Appendix C

TENA Core capabilities Details

At the most abstract level the invariant part of every instance of the Logical Range can be considered to consist of the system infrastructure services and mandatory system applications. This is called the TENA Core. It is represented graphically below. One of the most important aspects of the TENA Core is that it is standard across all instances of the Logical Range, with a minor exception. This exception is that some portions of the standard mandatory applications may be customized to suit particular range or facility circumstances when necessary. This is the only portion of the TENA Core that is considered "site-specific" with the remainder being entirely invariant. This site-specific portion of the otherwise invariant mandatory applications will be specified fully prior to implementation so that a clear distinction is made between the TENA Core invariant sections and the sections permitted to be customized.

The customization permitted to accommodate site uniqueness will not interfere with or alter the standardization of the interfaces or functioning of these mandatory applications. It will enable the sites to tailor the applications to conform to unique site configurations (hardware specific), procedures (site-specific operational requirements), and practices (site specific logs and record keeping).

This appendix provided the details of the TENA Core capabilities. It defines the services within each service group in terms of:

Description

In Parameters

Out Parameters

Exceptions

Pre Conditions

Post Conditions

It also provides details of how the services are used and how they interact by presenting Use Cases with Scenarios and Event Trace drawings whenever the service is complicated or when the use of a service causes other services to be involved in a complete transaction.

The TENA Core is composed of an integrated union of Service groups and mandatory applications that are standard across all instances of the Logical Range. The Services provided by the TENA Core have been logically grouped into 5 service groups of closely related services, and 4 required or mandatory applications. These service groups are:

- Distribution Services
- Message Services
- Connection Services
- Clock Services
- Infrastructure Support Objects

The required or mandatory applications are:

- Asset Manager
- Execution Manager
- Initialization Manager
- Network manager

These service groups are best understood when viewed as logical groupings of closely related service groups.

The first three service groups, Distribution, Message, and Connection Services, are responsible for the object based subscription service which provides for all information movement within the system, whether data or control. Distribution Services provides the only procedural interface within the infrastructure sevices. Access to all other assets including all other services of the infrastructure is via this object based subscription service.

The remaining two groups, Clock Services and Infrastructure Support Objects provide capabilities for coordinating time on the logical range and internal functioning of the infrastructure.

The four required applications, Asset Manager, Execution Manager, Initialization Manager, and Network manager are responsible for supporting the planning, scheduling, and execution of tests/exercises on instances of the logical range including managing any initialization data required.

It should be noted that there is a collection of tradeoffs (advantages, disadvantages) that were considered when determining whether a required capability of TENA should be defined as an infrastructure service or as a required application that compliments and completes the services. This current partitioning does not mean that some of the required applications could not be incorporated as a service of the infrastructure, only that it does not seem mandatory to do so. Several of the capabilities currently defined as applications were originally envisioned as infrastructure services, and may return to that category as more detailed information becomes available through future verification, validation, prototyping and testing of TENA.

Distribution Services:

Description:

Manages the distribution of data between range assets and infrastructure components. This service makes use of message services and connection services as appropriate to distribute the data between assets, and uses event classes to distribute information between infrastructure components. This component provides a subscription oriented service for data based on classes and class attributes. Data transfers are supported with a variety of quality of service factors on both shared and dedicated channels. Components instantiate the object classes they publish as required by their operation. At some appropriate time they register these instances with the range infrastructure by a service call to distribution services. Distribution services then advises all subscribers to the class of the existence of the instances through the discover service. This is one of the services that invokes a callback in a system asset.

Assumptions:

This system relies on an object based concept. Distribution of data throughout the system is aligned with this model of the system. System components establish classes that define the objects or instances of the classes from which the system is constructed. In terms of data transfer, distribution services expects that components requiring service provide definitions of classes of objects and classes of events. These class definitions contain attributes that are the carriers of the state information that is distributed within the system.

Instance of event classes (events) are created as required and dispatched into the system. There is no registration or discovery associated with these class instances. They are considered discrete, self contained, and non-persistent.

The minimum granularity to which distribution is managed is the attribute level. Components announce their intent to produce data for general consumption by using the publish service. The class and attributes are identified to the infrastructure as part of the service. Additional parameters are provided to define conditions of delivery and certain data characteristics. Consumers of data identify their needs by using the subscribe service, identifying the class and attributes desired and requirements for rate of delivery, etc.

It should be noted that the Quality of Service requested by the publisher of data must be prearranged with all planned subscribers since no mechanism is supported for modifying the QOS during operations. Different QOS required by different subscribers will be supported by separate classes with separate QOS being published .

Components are under no obligation to publish all of the information they contain. They announce availability via a publish service for only those classes and attributes they wish to make available outside the component. They can further specify access limitations on the data via parameters of the service. These access limitations establish subsets of components that can access a particular class. Distribution services enforces the access restrictions by checking a subscriber provided password against a publisher provided password. This password is arrived at outside of the Infrastructure so it can not be requested during an exercise. It must be established as a part of test scheduling when the all participants of an exercise are involved. The password protected data is limited on a class basis.

Rationale:

Standardization of communication within the system is critical. A new data or communication interface is accomplished by defining a Class and Attributes, then Publishing and Subscribing to them, with no new software required.

All Infrastructure components and all range assets will communicate through Distribution Services.

The intent of this concept is that Distribution Services is used by all system components and for interfaces within the system. All range assets that are not infrastructure assets accomplish all their communications through distribution services with rare exception. Infrastructure components use distribution services for all communications that travel between infrastructure instances. Because of this reliance of all other services on Distribution Service, it must be the first infrastructure service activated. It must also be self-initializing (i.e., no communication with Initialization Manager which is not yet active).

Services Provided:

<u>Generate Asset ID</u>

Description:

This service is used to generate a system unique identifier for an asset. This would be used prior to adding an asset to the master catalog.

In Parameters

- None

Out Parameters

- AssetID : Asset_Identifier - the new asset identifier requested..

Exceptions

- None

Preconditions

- None

Postconditions

- The generated asset identifier cannot be regenerated subsequent to additional invocations of this service.

Publish

Description:

A service used to announce the intention or capability of providing information described by parameters for use by system components.

In Parameters

- AssetID : Asset_Identifier - the identifier of the service requestor.
- Class : Class_Descriptor - the descriptor of the class that is being published.
- AttributeList : <sequence>Attribute_Descriptor - a list of attributes that are members of the class that the service requester is publishing.
- QualityOfService : <sequence>Quality_Of_Service_Value - a list of the quality of service parameters of importance and their values that are being requested for delivery of the class attribute updates.

- GroupID : Access_Group_Identifier - an identifier for an access group. This is an optional parameter that is required if distribution is not public. Subscribers must provide the correct group ID and password to receive any information from the publishing source.

- GroupPassword : Access_Group_Password - an optional parameter that is required when the class information distribution is restricted. Subscriber provided passwords are checked against this password when determining if access is authorized. The password is arrived at and distributed outside of the infrastructure so that only those parties publishing and subscribing to a particular class have access to it when this is required.

Out Parameters

- DeliveryHandle : Method_Handle - the handle to be used for dispatching of update and send event services. This is basically a method address provided to the publisher to use when executing the update and send event services.

Exceptions

- InvalidClass - the class descriptor provided is not valid; it could not be found in the information class catalog.

- InvalidAssetId – the Asset Id provided is not valid.

- InvalidAttribute - An attribute descriptor was not valid for the class identified.

- AlreadyPublished – the class is already being published.

- InvalidGroupId – the GroupID provided is not a valid value for the type specified (Implementation specific).

•Preconditions

- The class provided is defined in the information class catalog.

- The attributes are associated with the class in the information class catalog.

Postconditions

- Range infrastructure instances have recorded the asset as publishing the class and attributes indicated.

Unpublish

Description:

Used by a system component to announce its intent to cease providing the information for the selected class described by the parameters. All instances of the Class which were registered must first be removed before executing this service.

In Parameters

- AssetID : Asset_Identifier - the identifier of the service requestor.
- Class : Class_Descriptor - the descriptor of the class that is being published.

Out Parameters

none.

Exceptions

- StillRegistered – At least one instance is still registered.
- NotPublished - the class indicated or one or more of the attributes indicated are not currently being published by the service requestor.
- InvalidClass - the class descriptor provided is not valid; it could not be found in the information class catalog.
- InvalidAssetId – the Asset Id provided is not valid.

Preconditions

- The requester is publishing the class and attributes indicated.
- All instances of the class are deleted.

Postconditions

- Register service calls for instances of the class from the requester of this service are ignored.
- All updates or send event service requests for this class are ignored.

RegisterInstance

Description:

This service is used by a system component to announce the existence of an instance of some class that it publishes. This establishes the intent to provide state data about this instance.

In Parameters

- Class : Class_Descriptor - the descriptor of the class of the instance.
- AssetID : Asset_Identifier - the identifier of the service requestor.

Out Parameters

- InstanceID : Instance_Descriptor - the unique descriptor for the class instance.

Exceptions

- NotPublished – the Class is not being published.
- InvalidAssetID - the asset identifier supplied is not valid.
- InvalidClass - the class descriptor provided is not valid; it could not be found in the information class catalog.

Preconditions

- The requester is publishing the class.

Postconditions

- The requester may issue updates for that instance.
- All subscribers of the class are issued a DiscoverInstance for the instance.

Discover Instance

Description:

This service is an infrastructure initiated service that is invoked for all subscribers to a class when an instance of the class has been registered.

In Parameters

- InstanceID : Instance_Descriptor – the instamce identifier for the discovered instance.
- Class : Class_Descriptor - the descriptor of the class of the instance.

Out Parameters

none.

Exceptions

- None

Preconditions

- An asset is subscribed to the class and at least one of its attributes.

Postconditions

- The receiving asset will recognize updates for the class instance.

Update

Description:

Used by publishers of information to provide values for the indicated attributes of the indicated Class instance. Update services propagates the values provided to all components that have expressed a desire to receive the information provided that the discovery criteria for the receiver has been met and the information passes through any filters established in the delivery channel.

When a Class is published, Distribution Service provides a DeliveryHandle to be associated with the future Updates of the Class instance. Although the DeliveryHandle is not passed as a parameter for the Update service, it is used by the updating asset in directing or delivering the update in accordance with the syntax of the programming language being used. The API for Distribution Service will contain specific information on the use of the DeliveryHandle. The use of the

DeliveryHandle permits Distribution Services to set up entries to lower levels in the stack of infrastructure services to allow lower latency updates over dedicated public channels through Connection Services.

In Parameters

- InstanceID : Instance_Descriptor - the identity of the class instance that the update pertains to.
- AttributeValueList : <sequence>Attribute_Value_Record. A list containing pairs (Attribute Id and value) representing the identity of an attribute and its current value.

Out Parameters

none.

Exceptions

- NotPublished - one or more of the attributes listed or the class is not published.
- InvalidAttribute - an attribute descriptor was not valid for the class identified.
- InvalidInstanceID - the instance ID provided is not valid.

Preconditions

- The class and attributes are published.
- The class instance has been registered.

Postconditions

- The infrastructure propagates the update to subscribed assets.

SendEvent

Description:

Used by a component to dispatch a discrete event into the system. The event will

be propagated to all indicated components that have expressed a desire to receive the event.

In Parameters

- Class : Class_Descriptor - the descriptor of the event class.
- AssetID : Asset_Identifier - the identifier of the service requestor.
- AttributeValueList: <sequence>Attribute_Value_Record. A list containing pairs of values representing the identity of an attribute and its value.
- Destination : Asset_Identifier - the identifier of the asset that is the directed destination of this event. This is an optional parameter. If it is not present the event is sent to all assets that have subscribed to the class. If used it indicates that the event is to be directed to the indicated asset and only to that asset.

Out Parameters

none.

Exceptions

- NotPublished - one or more of the attributes listed or the class is not published.
- InvalidAssetId – the Asset Id provided is not valid.
- InvalidClass – the Class provided is not valid.
- InvalidAttribute - an attribute descriptor was not valid for the class identified.
- InvalidDestination - the destination is not a valid asset identifier.
- Not EventClass - the class of the event is not an event type class.
- DestinationNotSubscribed - the directed destination is not subscribed to the event class.

Preconditions

- The requester is publishing the event class.
- The requester is publishing the indicated attributes.
- The destination is subscribed to the class (when used).

Postconditions

- The event is propagated to the destination or assets subscribed to the event class.

Stop Updating

Description:

This service is used to advise the publisher to cease updating a class that it is publishing.

In Parameters

- Class : Class_Descriptor - the descriptor of the class published.

Out Parameters

- None.

Exceptions

- NotPublished – the Class is not being published.
- InvalidClass - the class descriptor provided is not valid; it could not be found in the information class catalog.

Preconditions

- The class is being published and an instance has been registered.
- There are no current subscribers to this class.

Postconditions

- The publisher ceases to update information for that class.

Resume Updating

Description:

This service is used to notify the publisher of a class to resume updating.

In Parameters

- Class : Class_Descriptor - the descriptor of the class .

Out Parameters

- None.

Exceptions

- InvalidClass - the class descriptor provided is not valid; it could not be found in the information class catalog.

Preconditions

- The class is being published and an instance has been registered.

Postconditions

- The publisher may resume issuing updates for that class instance.

Unregister Instance

Description:

This service is invoked to announce that an owner of a class instance will cease to provide information about that instance. It is propagated to subscribers to a class to advise them that an instance of the class has been removed from the system. This indicates that no more information about this instance will be delivered.

In Parameters

- InstanceID : Instance_Descriptor – the instance to be removed.
- AssetID : Asset_Identifier - the identifier of the service requestor.

Out Parameters

none.

Exceptions

- InvalidInstanceID - the instance ID provided is not valid.

- UnknownInstance - the instance indicated has not been discovered by the receiving asset.
- InvalidAssetId – the Asset Id provided is not valid.

- Preconditions
  - The class instance is registered.

  Postconditions
  - The class instance ceases to be recognized by the system.

Subscribe

Description:

A service used by any component that wishes to announce a need for information described in the parameters. Subscribers are notified of the registration of all class instances that meet the discovery criteria specified in this service.

In Parameters

- AssetID : Asset_Identifier - the identifier of the service requestor.
- Class : Class_Descriptor - the descriptor of the class that is being subscribed to.
- AttributeList : <sequence>Attribute_Descriptor - a list of attributes that are members of the class that the service requester is interested in receiving.
- QualityOfService : <sequence>Quality_Of_Service_Value - a list of the quality of service parameters of importance and their values that are being requested for delivery of the class attribute updates. This list includes an optional circuit descriptor that is required when the circuit connection is scheduled to be a dedicated public circuit rather than a general public circuit. The updates are propagated through Message services when a general public circuit is used, and through Connection services when a dedicated public circuit is used.

- GroupID : Access_Group_Identifier - an identifier for an access group. Optional parameter that is required if distribution is not public.
- GroupPassword : Access_Group_Password - an optional parameter that is required when the class information distribution is restricted. This is checked against the publisher supplied password and must agree if subscriber is to receive the requested data.

Out Parameters

none.

Exceptions

- InvalidClass - the class descriptor provided is not valid; it could not be found in the information class catalog.
- InvalidAttribute - An attribute descriptor was not valid for the class identified.
- InvalidAssetId – the Asset identifier is not valid.
- InvalidGroupID - the group ID provided is an invalid ID.
- PasswordFailure - the password provided is not correct for the access group indicated or a password was not provided when required.

Preconditions

- The class provided is defined in the information class catalog.
- The attributes are associated with the class in the information class catalog.

Postconditions

- The subscriber receives notification of registration of instances of the class.
- The subscriber receives updates or events for the subscribed class.

Unsubscribe

Description:

Used by system components to remove themselves from the distribution lists for the specified information.

In Parameters

- AssetID : Asset_Identifier - the identifier of the service requestor.
- Class : Class_Descriptor - the descriptor of a class the service requester is currently subscribed to and that it wishes to cease subscribing to.

Out Parameters

none.

Exceptions

- InvalidClass - the class descriptor provided is not valid; it could not be found in the information class catalog.
- InvalidAssetId – the Asset identifier is not valid.
- NotSubscribed - the class indicated or one or more of the attributes indicated are not currently subscribed to.

Preconditions

- The requester is subscribed to the class and attributes indicated in the parameters.

Postconditions

- The requester is no longer subscribed to the class.
- The subscriber no longer receives notice of discovery of any class instance.

ActivateFilter

Description:

The use of Publish and Subscribe services creates information channels within the system. Each pairing of a publisher with a subscriber defines one of these logical channels. For many reasons related to performance of communications networks, resource contention at devices with service network connections, cost

of network bandwidth, etc. there is a need to limit information transfer.

The establishing of interest groups via Publish and Subscribe is one form of limitation of transfer since the services establish a minimal connection network among components. However, many cases require additional filtering on information channels.

This service is used to install a filter in an information channel to effect a desired degree of limitation. It is the subscriber to data who installs a filter to protect itself from being overwhelmed by information on the channel. There is no inherent limitation on the number or type of filters that may be installed in a channel but implementations will probably constrain this variability.

The filter can be thought of as a gate through which information is required to flow. Information that fails to satisfy the gate's criteria closes the gate. The location at which this filtering occurs is determined by a parameter provided when the filter is activated. In general, filtering is performed most effectively at the source of the information but may be installed at the destination if multiple subscribers prohibits filtering at the source. This decision is made at test planning and is known when the request to activate the filter is issued.

In Parameters

- AssetID : Asset_Identifier - the identifier of the service requestor.
- FilterType : Filter_Descriptor - an indicator specifying which filter type is to be used.
- Class : Class_Descriptor - the class that the filter is to be applied to.
- FilterParameters : <sequence>Filter_Parameter_Value - a list of pairs of filter parameter identifiers and a value for each of those. Includes site where filter is to be installed.

Out Parameters

none.

Exceptions

- InvalidAssetId – the Asset identifier is not valid.
- InvalidFilterType - the filter type requested is not a valid choice.
- InvalidClassDescriptor - the class indicated is not valid.
- ClassNotSubscribed - the asset is not subscribed to the indicated class.
- InvalidParameter - one or more of the filter parameters or values were not valid.

- MissingParameter – A required filter parameter was not provided.

Preconditions

- The asset is subscribed to the indicated class.

Postconditions

- Updates from the indicated class are limited to those that pass the filter.

ModifyFilter

Description:

This service is used to change the value of one or more parameters to a filter that is installed in an information channel.

In Parameters

- AssetID : Asset_Identifier - the identifier of the service requestor.
- FilterType : Filter_Descriptor - an indicator specifying which filter type is to be used.
- Class : Class_Descriptor - the class that the filter is to be applied to.
- FilterParameters : <sequence>Filter_Parameter_Value - a list of paris of filter parameter identifiers and a value for each of those.

Out Parameters

none.

Exceptions

- InvalidAssetId – the Asset identifier is not valid.
- InvalidFilterType - the filter type requested is not a valid choice.
- InvalidClassDescriptor - the class indicated is not valid.
- ClassNotSubscribed - the asset is not subscribed to the indicated class.
- FilterNotActive - the type filter indicated was not active for the class indicated.

- InvalidParameter - one or more of the filter parameters or values were not valid.

Preconditions

- The indicated class is being subscribed to.
- There is a filter of the indicated type active for the class indicated.

Postconditions

- The filter operates according to the new parameter values.

DeactivateFilter

Description:

This service is used by a component that has established an information filter in one of its information channels to deactivate that filter. If there are multiple filters installed in a channel they must be deactivated individually.

In Parameters

- AssetID : Asset_Identifier - the identifier of the service requestor.
- FilterType : Filter_Descriptor - an indicator specifying which filter type is to be used.
- Class : Class_Descriptor - the class that the filter is to be applied to.

Out Parameters

none.

Exceptions

- InvalidAssetId – the Asset identifier is not valid.
- InvalidFilterType - the filter type requested is not a valid choice.
- InvalidClassDescriptor - the class indicated is not valid.
- FilterNotActive - the type filter indicated was not active for the class indicated.

Preconditions

- The indicated class is being subscribed to.
- There is a filter of the indicated type installed for the class indicated.

Postconditions

- The filter indicated for the specified class is deactivated.

RequestUpdate

Description:

This service can be used by a system component that has subscribed to some information that is being published by some other system component to query the publisher for the current value of the information described by the parameters. This applies to static classes and not events.This request has optional parameters that can be used to direct the request to an particular asset or apply to a specific class instance. If no optional parameters are used the request is directed to all assets that are publishing the class. For all instances the response to this query is satisfied by publishers issuing updates for the indicated information. These updates are directed to the specific requesters.

In Parameters

- AssetID : Asset_Identifier - the identifier of the service requestor.
- Class : Class_Descriptor - the descriptor of the class that the user requires data from.
- AttributeList : <sequence>Attribute_Descriptor - a list of attributes that are properties of the class of which the service requester desires current values.
- InstanceID : Instance_Descriptor - an optional parameter. When present it indicates the identity of the class instance for which the update is being requested.
- PublishAssetId : Asset_Identifier – (optional parameter) the identifier of the asset the request is directed to.

Out Parameters

- AttributeValueList : <sequence>Attribute_Value_Record. A list containing

pairs (Attribute Id and value) representing the identity of an attribute and its current value.

Exceptions

- InvalidAssetID - the asset identifier supplied is not valid.
- InvalidClass - the class descriptor provided is not valid; it could not be found in the information class catalog.
- InvalidAttribute - an attribute descriptor was not valid for the class identified.
- InvalidInstanceID - the instance ID provided is not valid (if used).
- InvalidPublishAssetID - the asset identifier supplied is not valid (if used).

Preconditions

- The class is being published.
- The attributes are being published.
- An instance is registered (as parameters indicate).
- Values have been assigned to the instance attributes.

Postconditions

- Updates are generated for the indicated class instances and attributes, and directed to the requestor.

RegisterChannel

Description:

This service is used to identify the protocol and circuit that has been allocated as a dedicated public circuit. This service is used to identify the actual circuit descriptor that is to be used in propagating the updates for the published class.

In Parameters

- ChannelID : the identifier of the channel.
- CircuitDescriptor : the descriptor of the protocol that is being registered.

Out Parameters

none.

Exceptions

- InvalidDescriptor - the descriptor provided is not valid.

Preconditions

- The circuit has been allocated.

Postconditions

- Distribution Service recognizes the circuit is supported by Connection Service.

UnegisterChannel

Description:

This service is used to indicate that the protocol and circuit that has previously been allocated as a dedicated public circuit is no longer required. This service is used to identify the actual circuit descriptor that is to be removed from the internal tables of Distribution Services.

In Parameters

- ChannelID : the identifier of the channel..

Out Parameters

none.

Exceptions

- InvalidChannel - the descriptor provided is not valid.
- NotRegistered – the channel has not been registered.

Preconditions

- The channel has been registered.

Postconditions

- Distribution Service recognizes the circuit is no longer supported by Connection Service.

TerminateAsset

Description:

This service is used to clean up the tables maintained by Distribution service when an asset has been placed out of service by a user command to Execution Manager. This is similar to the clean up that is normally performed at the end of an exercise or when an asset removes itself from the exercise. In this case the asset has failed and is not able to participate in the normal termination sequence.

In Parameters

- AssetId : the identification of service requestor.
- TargetId : the identification of the asset being terminated.

- Out Parameters

  none.

  Exceptions

  - InvalidAssetID - the asset identifier supplied is not valid.
  - UnknownAsset- the target asset is unknown.

  Preconditions

  - The asset is participating in an exercise and is known to distribution services.

  Postconditions

  - All publications and subscriptions by the asset are removed from the system.
  - All Class instances registered by the asset are removed.

Interfaces:

Use Cases:

It should be noted that the following use cases and event traces were developed at a stage when Asset Manager, Execution Manager, and Initialization Manager were considered Service Groups within the Infrastructure and not as required

applications supporting the infrastructure. Although these use cases are presented in their original form, the reader should be aware that updated versions depicting these three capabilities as applications would differ from this original depiction.

Publish and Update Class Use Case

Description:

This use case describes the publishing of a class by an asset, subscription to the class by assets, and subsequent registration of an instance with updates to attribute values. The use case pertains to a system consisting of three assets that are connected via local area networks and a wide area network. Both asset A and asset B are connected at a single facility via a local area network (LAN1). They are connected via a wide area network (WAN) through gateways G1 and G2 to asset C which connects to G2 via LAN2.

Assumptions:

a) The implementation of the network technology supports unicast and broadcast messaging. Implementers have chosen to not use multicast technology to support the system. This means that when there is a single subscriber to a class the sending infrastructure (for update) sends the update message to the address of the destination asset. If 2 or more assets subscribe the update is sent as a broadcast message.

b) The implementation of the range infrastructure allocates the task of extracting the set of attributes which a subscriber included in its subscription request from the attribute set in an update message to the instance of the infrastructure located at the destination (subscriber) end of the information channel.

c) The quality of service requested for class X can be satisfied by using the IP/UDP protocol set and the shared local area network/wide area network.

Actors:

a) Asset A: An asset which publishes a class X with attributes {x1, x2, x3}.

b) Asset B: An asset which subscribes to class X, attributes {x1, x2}.

c) Asset C: An asset which subscribes to class X, attributes (x1,x2,x3}.

d) Distribution Services 1(DS1): The distribution services component of the infrastructure at asset A.

e) Message Services 1(MS1): The message services component of the infrastructure at asset A. Interfaces to and processing of operating system components beneath the infrastructure are encapsulated within this actor for clarity.

f) LAN1: The local area network connecting the facility including asset A and asset B.

g) Distribution Services 2 (DS2): The distribution services component of the infrastructure at asset B.

h) Message Services 2 (MS2): The message services component of the infrastructure at asset C. Interfaces to and processing of operating system components beneath the infrastructure are encapsulated within this actor for clarity.

i) Gateway 1 (G1): A gateway to the WAN installed at the facility connected by LAN1 and consisting of asset A and asset B (at least).

j) Gateway 2 (G2): A gateway to the WAN installed at the facility connected by LAN2 and consisting of asset C (at least).

k) Distribution Services 3 (DS3): The distribution services component of the infrastructure at asset C.

l) Message Services 3 (MS3): The message services component of the infrastructure at asset C. Interfaces to and processing of operating system components beneath the infrastructure are encapsulated within this actor for clarity.

m) LAN2: The local area network connecting the facility including asset C.

Scenario:

a. Asset A decides to publish class X with an attribute set consisting of {x1,x2,x3}. It executes a publish service request to distribution services in its local infrastructure instance (DS1).

b. DS1 processes the publish request and creates structures to manage the

routing of updates to subscribers for class X.

c. Asset B decides to subscribe to attributes x1 and x2 for class X. It issues a subscribe service request with its local infrastructure instance (DS2).

d. DS2 records the subscription information and packages a subscription message with a broadcast destination, a source of B and identity of the class and attributes requested. It dispatches this message by a call to MS2 to send the packet.

e. MS2 takes the packet, prepares a call to the protocol service stack, then executes the call, dispatching the message. The underlying protocols and operating system take the message, perform additional required processing and dispatch it to LAN1.

f. All stations on LAN1 respond to the broadcast message and intercept it from the network.

g. MS1 gets a callback from the underlying protocol after the message has been processed through the network interface unit, operating system, and protocols. The callback transmits the subscription message from B for class X.

h. MS1 calls DS1 to pass it the subscription message. DS1 records the subscription data and establishes a routing table entry for the class, attributes, and destination.

i. Simultaneously, gateway 1 has picked up the same subscription message and broadcasts it unto the WAN where G2 picks the message up.

j. G2 repeats the broadcast message unto LAN2.

k. The network interface unit at Asset C intercepts the subscription message, processes it through the operating system and protocols, and calls MS3 to deliver the subscription from B for class X.

l. MS3 processes the callback and calls DS3 to pass the message to it.

m. DS3 processes the message, caching the information for later use since asset C is not currently publishing class X.

n. Asset A registers an instance of class X by invoking the register service at DS1.

o. DS1 determines the instance ID (XA) and responds back to Asset A with the instance ID for the instance just registered.

p. DS1 determines that asset B is the only asset subscribing to class X and prepares a discover message for instance XA. That message is dispatched to address B by calling MS1 to send the packet.

q. MS1 processes the packet and dispatches the message to the underlying

protocol and thence unto LAN1.

r. The network interface unit at asset B recognizes its address and copies the message. It processes the message and passes it up the protocol stack to the MS2 callback.

s. MS2 processes the discover message and calls DS2 to pass the message up.

t. DS2 processes the message and records pertinent information. It the calls asset B to deliver the discover message for class X instance XA.

u. Asset A issues an update for instance XA providing the current value for the attributes x1, x2, and x3. It invokes DS1's update service.

v. DS1 determines that asset B is the only subscriber to class X and its requested attribute set intersects the update attribute set. It prepares an update message with destination of B and dispatches it with a call to MS1's send service.

w. MS1 processes the update message passing it to the underlying protocol. It gets passed through to LAN1.

x. Asset B picks up the message off of LAN1 processing it through the OS and protocols whence it arrives at MS2 via a callback.

y. MS2 processes the update message and calls DS2.

z. DS2 extracts attributes x1 and x2 then passes the update to asset B.

aa. Asset C decides to subscribe to class X and attributes x1, x2, and x3. It prepares a request and invoked DS3's subscription service.

ab. DS3 processes the subscription, recording pertinent information, and prepares a subscription message to be broadcast. It calls MS3 to dispatch the subscription message.

ac. MS3 processes the message and passes it down the stack to LAN2.

ad. G2 picks up the broadcast message off of LAN2 and relays it through the WAN to G1.

ae. G1 broadcasts the message unto LAN1.

af. The network interface units at asset A and asset B both respond to the broadcast message, passing it up through the stack to their respective message services.

ag. MS1 processes the message and calls DS1 with the subscription message.

ah. MS2 also processes the message and calls DS2 with the same subscription message.

ai. DS1 processes the subscription and enters asset C into the distribution set for class X.

aj. DS2 caches the subscription information for later use but takes no further action since asset B is not currently publishing class X.

ak. DS1 completes its processing and dispatches a discover message to asset C since it has an instance (XA) currently registered. It builds the message and calls MS1 to dispatch it.

al. MS1 processes the message and passes it down the stack and out unto LAN1 with a destination address of C.

am. G1 responds to the message since destination C is in its map table and marked to pass. It relays the message through the WAN to G2.

an. G2 transmits the message to C over LAN2.

ao. MS3 receives the message after it has been received and processed up the stack. It processes the message and calls DS3 to deliver it.

ap. DS3 processes the message and calls asset C through a callback to pass the discover message to it.

aq. Asset A issues an update for instance XA providing the current value for the attributes x1, x2, and x3. It invokes DS1's update service.

ar. DS1 determines that there are multiple subscribers to class X whose requested attribute sets intersect the update attribute set. It prepares an update message with a broadcast destination and dispatches it with a call to MS1's send service.

as. MS1 processes the update message passing it to the underlying protocol. It gets passed through to LAN1.

at. G1 responds to the broadcast message by relaying it over the WAN to G2.

au. Asset B picks up the message off of LAN1 processing it through the OS and protocols whence it arrives at MS2 via a callback.

av. G2 broadcasts the update message unto LAN2 where it is picked up by asset C. The message is processed through the stack and passed up to MS3.

aw. MS2 processes the update message and calls DS2.

ax. MS3 processes the message and calls DS3

ay. DS2 extracts attributes x1 and x2 then passes the update to asset B.

az. DS3 extracts attributes x1, x2, and x3 then passes the update to asset C.

Event Trace:

Publish and Send Event Use Case

Description:

This use case describes the publishing of an event class by two assets, subscription to the event classes by assets, and the sending of events during the course of operations. The use case pertains to a system consisting of three

assets which are connected via local area networks and a wide area network. Both asset A and asset B are connected at a single facility via a local area network (LAN1). They are connected via a wide area network (WAN) through gateways G1 and G2 to asset C which connects to G2 via LAN2.

Assumptions:

The implementation of the network technology supports unicast and broadcast messaging. Implementers have chosen to not use multicast technology to support the system. This means that when there is a single subscriber to a class the sending infrastructure sends the event to the address of the destination asset. If 2 or more assets subscribe the event is sent as a broadcast message.

The implementation of the range infrastructure allocates the task of extracting the set of attributes which a subscriber included in its subscription request from the attribute set in an event message to the instance of the infrastructure located at the destination (subscriber) end of the information channel.

The quality of service requested for class X can be satisfied by using the IP/UDP protocol set and the shared local area network/wide area network.

Actors:

a) Asset A: An asset which publishes event class X with attributes {x1, x2, x3}.

b) Asset B: An asset which publishes event class Y with attributes {y1,y2} and subscribes to event class X, attributes {x1, x2}.

c) Asset C: An asset which subscribes to event class Y, attributes (y1,y2}.

d) Distribution Services 1(DS1): The distribution services component of the infrastructure at asset A.

e) Message Services 1(MS1): The message services component of the infrastructure at asset A. Interfaces to and processing of operating

system components beneath the infrastructure are encapsulated within this actor for clarity.

f) LAN1: The local area network connecting the facility including asset A and asset B.

g) Distribution Services 2 (DS2): The distribution services component of the infrastructure at asset B.

h) Message Services 2 (MS2): The message services component of the infrastructure at asset C. Interfaces to and processing of operating system components beneath the infrastructure are encapsulated within this actor for clarity.

i) Gateway 1 (G1): A gateway to the WAN installed at the facility connected by LAN1 and consisting of asset A and asset B (at least).

j) Gateway 2 (G2): A gateway to the WAN installed at the facility connected by LAN2 and consisting of asset C (at least).

k) Distribution Services 3 (DS3): The distribution services component of the infrastructure at asset C.

l) Message Services 3 (MS3): The message services component of the infrastructure at asset C. Interfaces to and processing of operating system components beneath the infrastructure are encapsulated within this actor for clarity.

m) LAN2: The local area network connecting the facility including asset C.

Scenario:

a. Asset A decides to publish event class X with an attribute set consisting of {x1,x2,x3}. It executes a publish service request to distribution services in its local infrastructure instance (DS1).

b. DS1 processes the publish request and creates structures to manage the routing of events to subscribers for class X.

c. Asset B publishes event class Y with attributes {y1,y2}. It executes a publish service request to distribution services in its local infrastructure instance (DS2).

d. DS2 processes the publish request and creates structures to manage the routing of events to subscribers for class Y.

e. Asset B decides to subscribe to event class X, and attributes x1 and x2. It issues a subscribe service request with its local infrastructure instance (DS2).

f. DS2 records the subscription information and packages a subscription

message with a broadcast destination, a source of B and identity of the class and attributes requested. It dispatches this message by a call to MS2 to send the packet.

g. MS2 takes the packet, prepares a call to the protocol service stack, then executes the call, dispatching the message. The underlying protocols and operating system take the message, perform additional required processing and dispatch it to LAN1.

h. All stations on LAN1 respond to the broadcast message and intercept it from the network.

i. MS1 gets a callback from the underlying protocol after the message has been processed through the network interface unit, operating system, and protocols. The callback transmits the subscription message from B for class X.

j. Simultaneously, gateway 1 has picked up the same subscription message and broadcasts it unto the WAN where G2 picks the message up.

k. MS1 calls DS1 to pass it the subscription message. DS1 records the subscription data and establishes a routing table entry for the class, attributes, and destination.

l. G2 repeats the broadcast message unto LAN2.

m. The network interface unit at Asset C intercepts the subscription message, processes it through the operating system and protocols, and calls MS3 to deliver the subscription from B for class X.

n. MS3 processes the callback and calls DS3 to pass the message to it.

o. DS3 processes the message, caching the information for later use since asset C is not currently publishing class X.

p.

q. Asset C decides to subscribe to class Y and attributes y1 and y2. It prepares a request and invoked DS3's subscription service.

r. DS3 processes the subscription, recording pertinent information, and prepares a subscription message to be broadcast. It calls MS3 to dispatch the subscription message.

s. MS3 processes the message and passes it down the stack to LAN2.

t. G2 picks up the broadcast message off of LAN2 and relays it through the WAN to G1.

u. G1 broadcasts the message unto LAN1.

v. The network interface units at asset A and asset B both respond to the

broadcast message, passing it up through the stack to their respective message services.

w. MS1 processes the message and calls DS1 with the subscription message.

x. MS2 also processes the message and calls DS2 with the same subscription message.

y. DS2 processes the subscription and enters asset C into the distribution set for class Y.

z. DS1 caches the subscription information for later use but takes no further action since asset B is not currently publishing class Y.

aa. Asset B issues a send event for class Y providing the current value for the attributes y1 and y2. It invokes DS2's update service.

ab. DS2 determines that there is a single subscriber, C to class Y whose requested attribute sets intersect the update attribute set. It prepares an event message with a destination C and dispatches it with a call to MS2's send service.

ac. MS2 processes the update message passing it to the underlying protocol. It gets passed through to LAN1.

ad. G1 responds to the message by relaying it over the WAN to G2.

ae. G2 transmits the event message unto LAN2 where it is picked up by asset C. The message is processed through the stack and passed up to MS3.

af. MS3 processes the message and calls DS3

ag. DS3 extracts attributes y1 and y2 then passes the event to asset C.

ah. Asset A issues a send event service request for event class X with attribute values for x1, x2, and x3 to DS1.

ai. DS1 determines that there is a single subscriber for event class X, B, and packages the event up in a message. It dispatches the message with a call to MS1's send service.

aj. MS1 processes the message and passes it down the stack and out unto LAN1.

ak. Asset B's network interface unit intercepts the message destined for B from LAN1 and processes it passing it up to MS2.

al. MS2 processes the message and passes the send event packet to DS2.

am. DS2 extracts the values of attributes x1 and x2 then passes the event up to asset B.

Event Trace:

Unsubscribe, Stop, Subscribe, and Resume Use Case

Description:

This Use Case depicts the unsubscribing of a class by all assets that have previously subscribed to it, and a resulting STOP service issued to the publisher. To continue the sequence, a new subscribe is performed with a resulting RESUME service issued to the publisher. A DISCOVER is issued to the new

subscriber and an UPDATE is issued by the publisher

Assumptions:

This is a continuation of the previous Use Case with all assumptions maintained.

Actors:

The same Actors as in the previous Use case with the addition of Asset D which subscribes to Class X, attributes (x1,x2)

Scenario:

a) Asset B decides to unsubscribe to attributes x1 and x2 for class X. It issues an unsubscribe service request with its local infrastructure instance (DS2).

b) DS2 records the unsubscribe information and packages an unsubscribe message with a broadcast destination, a source of B and identity of the class and attributes. It dispatches this message by a call to MS2 to send the packet.

c) MS2 takes the packet, prepares a call to the protocol service stack, then executes the call, dispatching the message. The underlying protocols and operating system take the message, perform additional required processing and dispatch it to LAN1.

d) All stations on LAN1 respond to the broadcast message and intercept it from the network.

e) MS1 gets a callback from the underlying protocol after the message has been processed through the network interface unit, operating system, and protocols. The callback transmits the unsubscribe message from B for class X.

f) MS1 calls DS1 to pass it the unsubscribe message. DS1 records the unsubscribe data and removes the routing table entry for the class, attributes, and destination.

g) Simultaneously, gateway 1 has picked up the same unsubscribe message and broadcasts it unto the WAN where G2 picks the message up.

h) G2 repeats the broadcast message unto LAN2.

i) The network interface unit at Asset C intercepts the unsubscribe message, processes it through the operating system and protocols, and calls MS3 to deliver the unsubscribe from B for class X.

j) MS3 processes the callback and calls DS3 to pass the message to it.

k) DS3 processes the message and discards it.

l) Asset A issues updates which are now only directed to Asset C.

m) Asset C decides to unsubscribe to class X and attributes x1, x2, and x3. It prepares a request and invoked DS3's unsubscribe service.

n) DS3 processes the unsubscribe, recording pertinent information, and prepares an unsubscribe message to be broadcast. It calls MS3 to dispatch the unsubscribe message.

o) MS3 processes the message and passes it down the stack to LAN2.

p) G2 picks up the broadcast message from LAN2 and relays it through the WAN to G1.

q) G1 broadcasts the message to LAN1.

r) The network interface units at asset A and asset B both respond to the broadcast message, passing it up through the stack to their respective message services.

s) MS1 processes the message and calls DS1 with the unsubscribe message.

t) MS2 also processes the message and calls DS2 with the same unsubscribe message.

u) DS1 processes the unsubscribe and removes asset C from the distribution set for class X.

v) DS2 discards the message.

w) DS1 determines there are no subscribers to Class X and issues a Stop service message to Asset A to inhibit further updates.

x) Asset D decides to subscribe to class X and attributes x1, x2, and x3. It prepares a request and invokes DS4's subscription service.

y) DS4 processes the subscription, recording pertinent information, and prepares a subscription message to be broadcast. It calls MS4 to dispatch the subscription message.

z) MS4 processes the message and passes it down the stack to LAN2.

aa. G2 picks up the broadcast message from LAN2 and relays it through the WAN to G1.

ab. G1 broadcasts the message to LAN1.

ac. The network interface units at asset A and asset B both respond to the broadcast message, passing it up through the stack to their respective message services.

ad. MS1 processes the message and calls DS1 with the subscription message.

ae. MS2 also processes the message and calls DS2 with the same subscription message.

af. DS1 processes the subscription and enters asset D into the distribution set for class X.

ag. DS2 discards the message.

ah. DS1 recognizes that Asset A is not updating Class X but has previously Registered it, so DS1 issues a Discover message to Asset D via Message services. DS1 also issues a Resume service message to Asset A.

ai. Asset A issues an update for instance XD providing the current value for the attributes x1, x2, and x3. It invokes DS1's update service.

aj. DS1 determines that asset D is the only subscriber to class X and its requested attribute set intersects the update attribute set. It prepares an update message with destination of D and dispatches it with a call to MS1's send service.

ak. MS1 processes the update message passing it to the underlying protocol. It gets passed through to LAN1.

al. G1 responds to the broadcast message by relaying it over the WAN to G2.

am. G2 broadcasts the update message unto LAN2 where it is picked up by asset D. The message is processed through the stack and passed up to MS4.

an. MS4 processes the update message and calls DS4.

ao. DS4 extracts attributes x1, x2, and x3 then passes the update to asset D.

Event Trace:

Group ID and Password Use Case

Description:

This Use Case depicts the use of the Group ID and Password to prevent unauthorized access of data being published.

Assumptions:

The pretest planning and scheduling meetings have determined that a class of data that will be published during the exercise will be restricted from public access.

The planned publisher and subscriber of the data agree to a Group Id and Password that will be used in publishing and subscribing this class of data.

This use case involves only a single valid subscriber. It should be noted that multiple valid subscribers would either involve multiple directly routed updates (one per subscriber, high overhead) or a broadcast of the updates. In broadcast mode, the infrastructure of the subscriber would decide if its subscriber is valid and permitted to receive the update.

Actors:

Asset A is the publisher of the non-public class of data.

DS1 is the Distribution Service that is within the same instance of the infrastructure as Asset A.

MS1 is the Message Service that is within the same instance of the infrastructure as Asset A.

Asset B is the valid subscriber to the non-public class of data.

DS2 is the Distribution Service that is within the same instance of the infrastructure as Asset B.

MS2 is the Message Service that is within the same instance of the infrastructure as Asset B.

Asset C is a subscriber that is not intended to have access to the non-public class of data.

DS3 is the Distribution Service that is within the same instance of the infrastructure as Asset C.

MS3 is the Message Service that is within the same instance of the infrastructure as Asset C.

Scenario:

Asset A issues a Publish service request for the non-public class of data and includes the Group Id and Password.

DS1 retains the Group Id and Password for later comparison to any attempts to subscribe to this class of data.

Asset A issues a Register Instance service request to declare its readiness to publish this class of data and is returned an InstanceId.

Asset B issues a Subscribe service request for the non-public class of data and includes the Group Id and Password.

DS1 compares the Group Id and Password provided with the Subscribe service request and determines the subscriber is permitted to have access to the class of data.

DS3 saves the Subscribe service request anticipating a possible Publish from Asset C at some later time.

DS1 updates its internal routing tables to indicate Asset B is a valid subscriber to this instance of the non-public class of data.

DS1 issues a Discover Instance to Asset B containing the InstanceId for this class.

Asset A issues an Update service request which contains the data of the non-public class.

DS1 examines its routing tables and propagates the data of the Update service request to Asset B.

Asset C issues a Subscribe service request for the non-public class of data but does not include the Group Id and Password.

DS1 determines that a password is required for this class of data and returns an exception to Asset C indicating password failure.

DS2 discards Subscribe service request.

Asset A continues to provide updates to Asset B.

Event Trace:

Filtering – TLM Data filtered based on position of TLM emitter

Description:

This Use Case depicts the interaction of the Service Infrastructure relating to a Request for a Filter. The Filter will pass TLM data through the filter gate whenever the test vehicle emitting the TLM is within certain positional limits. The position of the vehicle is obtained through a separate tracking asset.

Assumptions:

a) Asset 1 (A1) is publishing Class TLM Stream.

b) Asset 2 (A2) is publishing Class Track.

c) Asset 3 (A3) is subscribing to Class TLM Data and Class Track.

d) A1 has instantiated TLM stream (InstanceID = Stream1).

e) A2 has instantiated Track (InstanceIds = Track1,Track2,Track3).

f) During Test Planning, the filter has been determined to be of type 'Region' with parameters

[X_0 = Track.Xpos,

X_1 = Track.Ypos,

X_2 = InstanceID,

N_0 = Extent.X = X_{min} X_{max},

N_1 = Extent.Y = Y_{min} Y_{max},

N_2 = Stream1]

g) A3 has Discovered Stream1, Track1, Track2, and Track3.

h) A3 has made correlation (user or application assisted) of emitter of interest with Track2.

i) Track2 is outside of defined extent values.

Actors:

Scenario:

Asset 3 activates filter with parameters

[X_0 = Track.Xpos,

X_1 = Track.Ypos(Track2),

X_2 = InstanceID(Track2),

E_0 = Extent.X = X_{min} X_{max},

E_1 = Extent.Y = Y_{min} Y_{max},

E_2 = Stream1]

[ActivateFilter:A1(parameters as above)]

Distribution Service 1 subscribes to Class Track being published by Asset 2.

[Subscribe:A2(Class=Track)]

Asset 2 provides updates to Class Track(Track1, Track2, Track3) with Track2 outside of extent values.

[Update:Track(Track1, Track2, Track3).

Asset 1 provides update to Class TLM Stream.

[Update:TLM Stream(Stream1)]

Distribution Service 1 receives Track2 and applies its values against the extent values of the filter. Since the Track2 values are outside of the extent values it does not pass the TLM Stream through the filter gate.

Asset 2 provides updates to Class Track(Track1, Track2, Track3) with Track2 inside of extent values.

[Update:Track(Track1, Track2, Track3).

Asset 1 provides update to Class TLM Stream.

[Update:TLM Stream(Stream1)]

Distribution Service 1 receives Track2 and applies its values against the extent values of the filter. Since the Track2 values are now within the extent values, it updates the latest packet of the TLM Stream to Asset 3. It is the responsibility of Asset 3 to time correlate the values of the TLM Stream with the Track data using the time stamps associated with those records.

[Update:TLM Stream(Data)].

Event Trace:

Filtering – Select Sample Rate Use Case

Description:

This Use Case depicts the interaction of the Service Infrastructure relating to a Request for a Filter. The Filter will pass data through the filter gate whenever a defined sample rate is satisfied.

Assumptions:

a) Asset 1 (A1) is publishing Class X(a,b,c,d).

b) Asset 2 (A2) is subscribing to Class X(a).

c) Asset 1 has registered an instance of a, b, c, d.

d) Asset 1 is updating a, b, c, d at a two per second rate.

e) It has been decided at Planning/Test scheduling that Asset 2 will activate a filter of type 'time sampling' installed at Distribution Service 1 with a period equal to one second.

Actors:

Scenario:

Asset 1 issues updates for X(a, b, c, d) at 2 per second rate.

Asset 2 receives updates for X(a) at a 2 per second rate.

Asset 2 activates filter to limit the update rate for X(a) to one per second.

[ActivateFilter:A1(Class X(a), Type = TimeSample, Period = 1 per second)]

Asset 1 continues to issue updates for a, b, c, d at 2 per second rate

Distribution Service 1 retains a count of updates received for X(a) and propagates the values to Asset 2 at a one per second rate.

Asset 2 receives updates X(a) at a one per second rate.

Event Trace:

Message Services:

Description:

Message services provides a form of communications based on a discrete, packetized data oriented service for any purposes applicable within range operations. It is expected that all communications for discrete data streams which do not have exceptional latency or capacity requirements or which are not restricted to specific dedicated channels for other purposes will be managed through message services. It provides connections to one or more networks to support these services by making use of communications facilities provided by the execution platform. Users of message services can select among different networks and protocol sets to match provided capabilities and characteristics with required quality of service. This service functions primarily to encapsulate the interfaces to the network services present on the platform. It isolates users from the details of how protocols operate, what system services must be used, and format and content of control data.

From the users perspective this service is a datagram service. However, the quality of service negotiated may establish virtual circuits where required to ensure meeting guarantees for delivery, restrictions on ordering, etc.

The basic strategy for sending messages is for the service requester to buffer the outgoing message and provide parameter values to determine destination and distribution characteristics. The sender then calls the send service. For incoming messages the service requester can poll for messages by using the receive service, with appropriate parameter values. Implementations may also allow an interrupt driven option which allows message services to call an entry provided by the service requester to deliver the message. With an interrupt driven approach message services will not buffer messages for users. Service requesters can switch between polled and interrupt driven service as required.

However, when switching from polled to interrupt driven the service requester is responsible for retrieving any messages remaining on the incoming message queue when the switch is effected.

Assumptions

- Users of Message Services understand the quality of service which can be guaranteed for each protocol which Message Services provides access to. Users provide any mapping between requested quality of service parameter values and protocols supported here.

Rationale

The primary drivers for identifying the Message Services partition are separation of concern and information hiding. Both of these are consequences of a need to limit the complexity of Distribution Services by removing the need to deal with the details of how datagrams are sent and which platform facilities to make use of in dispatching packetized data from Distribution Services. The requirement to provide packetized network connectivity is allocated to Message Services to relieve Distribution Services of those details.

In order to provide the capability to select different strategies for managing the servicing of incoming message streams Message Services is provided with the ability to support both synchronous and asynchronous message delivery on a protocol basis. Users (Distribution Services) can alternate between the approaches as required to meet changing needs and better match overhead costs and latency to user requirements.

Services Provided:

Send

Description:

A service used to send a message.

Assumptions:

none.

Rationale:

This service provides the means for dispatching messages into supported networks. It supports all available protocols and allows the sending of untyped data blocks of the indicated length. It provides all processing required to wrap the data and provide it along with appropriate control information when the underlying platform capability is invoked.

In Parameters

- Data : DS_Message_Pkt* - A pointer to the block of data which is to be transmitted.
- Protocol : Channel_Identifier - The identifier of the protocol to be used for dispatching the message.

Out Parameters

none.

Exceptions

- InvalidAssetID - the destination address provided is not a valid asset identifier.
- Unknownchannel- the descriptor provided indicates a protocol which is not available to message services or is an invalid descriptor.
- InvalidAddress - the pointer provided for the data is not a valid address.
- UnableToSend - the message could not be sent because of an internal fault or platform system failure.

Preconditions

- The destination address is known to the system's network protocols.
- The communications protocols and communications networks have been initialized.

Postconditions

- The data indicated has been wrapped in appropriate protocol packets and dispatched into the indicated network.

Receive

Description:

A service used to extract the next message on the incoming message queue for the indicated protocol.

Assumptions:

none.

Rationale:

This provides the general purpose message retrieval facility. It supports all available protocols.

In Parameters

- Protocol : Channel_Identifier - The identifier of the protocol whose queue is to be queried and popped.

Out Parameters

- Data : DS_Message_Pkt* - a pointer to the message. This parameter = NULL if no message was found.

Exceptions

- UnknownChannel - the descriptor provided is not available to message services or is an invalid descriptor.

Preconditions

none.

Postconditions

- The service requester holds the top most message on the indicated protocol's incoming message queue and that message has been removed from the queue.

RegisterCallback

Description:

This service is used to register a method for callback service when an incoming message is received. A callback is registered for each protocol set which can be used to transfer messages. Invocation of this service sets the retrieval mode for the protocol to interrupt driven. The mode defaults to polled when the infrastructure is initialized and remains so until this service is invoked.

Assumptions:

- The callback address provided in a valid entry point for an executable code module.
- The called method copies the message data before returning.

Rationale:

In order to support both a polled as well as an interrupt driven strategy for

managing incoming messages this service is provided to allow users to be interrupted by incoming messages. This provides a capability to respond quickly to messages minimizing response latency.

In Parameters

- Protocol : Channel_Identifier - The identifier of the protocol which is to execute the callback on receipt of a message.
- Method : Receive_Handle - the address reference for the method to be used as an interrupt service routine.

Out Parameters

none.

Exceptions

- UnknownChannel - the protocol indicated is not known to message services.
- InvalidHandle - the address provided for the callback method is not a valid method address.

Preconditions

- Protocol software is initialized.
- The method servicing incoming message interrupts is initialized.

Postconditions

- Messages using the protocol indicated will be passed to the service requester using the callback method as the interrupt service routine.

CancelCallback

Description:

This service is used to cancel a previously registered callback. It removes the callback method address for the protocol indicated and marks the protocol as polled.

Assumptions:

none.

Rationale:

This service is required to allow users to switch from an interrupt driven message

dynamically between modes to better match transient latency and response requirements during critical periods.

- registered to execute the callback on receipt of a message.

Out Parameters

Exceptions

- UnknownChannel - the protocol indicated is not known to message services.

  NotRegistered - there is no callback registered for the protocol indicated.

-

Postconditions

- The protocol is marked as polled and when subsequent messages are received they will be queued until the service requester invokes the

Interfaces:

Description:

Connection Services provides the system mechanisms for dynamically adding and removing communications networks and protocols. The service supported here is similar to that provided by Message Services for well-known platform supported communications networks but allows those communications facilities to be added and removed from the system in response to the needs of specific logical ranges and, further, allows the use of special purpose protocols which support peculiar communications needs or afford improved quality of service tailored to test requirements.

Connection Services accepts requests to install communications connections into the system including arranging requested protocols and connecting them into the data stream. The assembly of protocols and connections are referenced by a communications channel number assigned by Connection Services and the quality of service parameter value set which the connection supports. It advises Distribution Services of the channel it supports along with the quality of service supported by the channel and a method handle used for sending information so that Distribution Services can provide support to publishers and subscribers.

Assumptions:

Asset Manager has ascertained the accuracy and completeness of information about connections attached to the operating platform.

Rationale:

Connection Services was created in response to a need to simplify Distribution Services by removing from it the need to deal with the specifics of how to connect up a communications channel. It was also separated from Message Services to allow each service group to optimize their implementation for either statically configured connections and protocols or dynamically configured connections and protocols.

Services Provided:

The connection services provide two distinct groups of services, network protocol messaging services as are provided using message services and network connection management. The TENA relies on network manager to provide range specific network management.

<u>Send</u>

Description:

A service used to send a message.

Assumptions:

none.

Rationale:

This service provides the means for dispatching messages into supported networks. It supports all available protocols and connections and allows the sending of untyped data blocks of the indicated length. It provides all processing required to wrap the data and provide it along with appropriate control information when the underlying platform capability is invoked.

In Parameters

Data : DS_Message_Pkt* - A pointer to the block of data which is to be transmitted.

- Protocol : Channel_Identifier - The identifier of the connection to be used for dispatching the message.

Out Parameters

none.

Exceptions

- InvalidAssetID - the destination address provided is not a valid address.
- UnknownChannel - the descriptor provided indicates a connection which is not available to message services or is an invalid descriptor.
- InvalidAddress - the pointer provided for the data is not a valid address.
- UnableToSend - the message could not be sent because of an internal fault or platform system failure.

Preconditions

- The destination address is known to the system.
- The communications protocols and communications networks have been initialized.

- dispatched into the indicated network.

Receive

A service used to extract the next message on the incoming message queue for the indicated connection.

Assumptions:

none.

Rationale:

This provides the general purpose message retrieval facility. It supports all available connections and protocols.

In Parameters

- Protocol : Channel_Identifier - The identifier of the connection whose queue is to be queried and popped.

Out Parameters

- Data : DS_Message_Pkt* - a pointer to the message. This parameter = NULL if no message was found.

Exceptions

- UnknownChannel - the descriptor provided is not available to message services or is an invalid descriptor.

Preconditions

none.

Postconditions

- The service requester holds the top most message on the indicated connection's incoming message queue and that message has been removed from the queue.

RegisterCallback

Description:

This service is used to register a method for callback service when an incoming message is received. A callback is registered for each connection which can be used to transfer messages. Invocation of this service sets the retrieval mode for the connection to interrupt driven. The mode defaults to polled when the infrastructure is initialized and remains so until this service is invoked.

Assumptions:

- code module.
- The called method copies the message data before returning.

Rationale:

managing incoming messages this service is provided to allow users to be interrupted by incoming messages. This provides a capability to respond quickly

In Parameters

- Protocol : Channel_Identifier - The identifier of the connection which is to execute the callback on receipt of a message.

  Method : Receive_Handle - the address reference for the method to be

Out Parameters

none.

- services.
- InvalidHandle - the handle provided for the callback method is not a valid method address.

-
-

Postconditions

- Messages using connection indicated will be passed to the service requester using the callback method as the interrupt service routine.

CancelCallback

Description:

This service is used to cancel a previously registered callback. It removes the callback method address for the connection indicated and marks the connection as polled.

Assumptions:

none.

Rationale:

This service is required to allow users to switch from an interrupt driven message management scheme to a polled scheme. This supports users switching dynamically between modes to better match transient latency and response requirements during critical periods.

In Parameters

- Protocol : Channel_Identifier - The identifier of the connection which is registered to execute the callback on receipt of a message.

Out Parameters

none.

Exceptions

UnknownChannel - the connection indicated is not known to message services.

NotRegistered - there is no callback registered for the connection indicated.

Preconditions

- A callback method is registered for the connection indicated.

Postconditions

- The connection is marked as polled and when subsequent messages are received they will be queued until the service requester invokes the receive service.

InstallConnection

Description:

This service is used by infrastructure components to install a communications circuit which has been attached to the operating platform. The Network Manager creates, acquires, or allocates a circuit and attaches to an input/output channel on the local platform. Connection information supplied by the Network Manager about this connection is provided to Connection Services to locate and characterize the circuit.

Assumptions:

- Network manager has completed the network connection prior to the InstallConnection invocation.

Rationale:

The InstallConnection service has been provided to provide a means of describing a new communications connection to the system to allow its use by assets.

In Parameters

- ConnectionParameters : <sequence>Connection_Param_Value- a set of pairs of connection parameters and their value describing the nature of the connection. These parameters include a description of the quality of service supported.
- Protocol : <sequence> Protocol_Descriptor_Value - a set of pairs of parameters and their value describing information that specifies the protocol to be used to service the connection.

Out Parameters

- ConnectionStatus - a code indicating status of connection

Exceptions

- InvalidConnectionID - the channel identifier is invalid.
- InvalidConnectionParameter - one or more of the connection parameters or parameter values in invalid.
- InvalidProtocolParameter - one or more of the protocol parameters or parameter values is invalid.

Preconditions

- The circuit has been attached to the indicated input/output channel.
- Required protocols are available.

Postconditions

- The communications channel is formed and has been registered with distribution services.
- The communications channel has been initialized.
- Distribution Services and assets may invoke the published services for the indicated connection.

RemoveConnection

Description:

This service removes a connection from the set of connections that are supported by Connection Services.

Assumptions:

none

Rationale:

This service is required to allow the dynamic removal of connections from the system.

In Parameters

- ConnectionID : Connection_Descriptor - the connection identifier of the channel which is to be removed.

Out Parameters

- ConnectionStatus - a code indicating status of connection.

Exceptions

- InvalidConnectionID - the channel identifier is invalid.

Preconditions

- The connection has been installed.

Postconditions

- The connection is no longer available to the system.

Description:

This service is provided to enable the reinitialization of a connection and

Assumptions:

none.

This is required to allow Connection Services to provide the ability to recover from serious errors.

- channel which is to be reset.

Out Parameters

Exceptions:

- InvalidConnectionID - the connection identifier is invalid.

Preconditions

The connection has been established and installed.

-

Description:

This service forces all data that has been queued within protocols associated with the connection, within communications nodes, and within network interface units to be delivered to the connection end point. Additional information will not be accepted until all connection data has been "flushed" from buffers and stacks.

Assumptions:

none.

Rationale:

This service allows the user to force data transfer (output) at the termination of operation or as part of an error recovery strategy.

In Parameters

- ConnectionID - the connection identifier of the connection to be flushed.

Out Parameters

none.

Exceptions

- InvalidConnectionID - the connection identifier is invalid.

Preconditions

- The connection has been installed and initialized.

Postconditions

- All connection buffers and stacks are empty.

Interfaces:

Clock Services

Description

The clock service group manages clock issues for the range. It performs synchronization and time setting services as well as maintaining a global clock for exercises.

Assumptions

The specification of this services group is based on the following assumptions:

- The infrastructure has a method for reliably maintaining an internal wall clock such as GPS.

- needed by clock services.
- The variability in network latency times is sufficiently small that time service algorithms can be calibrated to remove the network effects.

Synchronization of time is an important requirement for many TENA assets. Examples include radars and other devices which time stamp data before it is

collected during a test or exercise can be useless.

GetCurrentTime

This service is utilized to get the current value of date and time for the range system global clock. This is a single synchronized date and time which is shared

Assumptions

The specification of this service is based on the assumption that network delays

clock time from clock services.

Rationale

internal wall clock.

In Parameters

Out Parameters

CurrentTime : Date_Time - the current date and time in zulu timezone.

none

Preconditions

The infrastructure has initialized its time and date state information.

Postconditions

none.

SetCurrentTime

Description

This service is used by a range management asset with sufficient authority to set the current value of the date and time maintained for the range system clock. For ranges with assets that use GPS to maintain their clock, the SetCurrentTime will have no effect. SetCurrentTime is used to set the time for those assets without a GPS capability.

Assumption

The specification of this service is based on the assumption that network delays are within a range will not exceed the threshold for accurately setting the wall clock time of other services or assets.

Rationale

This service is needed to enable clock services to synchronize the wall clocks of all services and assets within a range.

In Parameters

DateTime

- Date_Time - the value of date and time in zulu timezone which the current date and time are to be set to.

Out Parameters

none

Exceptions

InvalidDateTime - the provided value for date and time is not a valid date and/or time.

none.

Postconditions

Description

This service is used to cause the infrastructure to propagate a date and time

Assumption

The specification of this service is based on the assumption that network delays

wall clocks of all services and assets within a range.

Rationale

are synchronized to the same time to reliably interpreting the results of a test, exercised, or event.

none

Out Parameters

Exceptions

AssetNotResponding - a known asset did not respond to the synchronization

ExtremeTimeDrift - the requested sync time differs greatly from currently maintained time.

none

Postconditions

- All assets have set their date and time to the same (zulu) values.

Interfaces:

Use Cases

It should be noted that the following use cases and event traces were developed at a stage when Asset Manager, Execution Manager, and Initialization Manager were considered Service Groups within the Infrastructure and not as required applications supporting the infrastructure. Although these use cases are presented in their original form, the reader should be aware that updated versions depicting these three capabilities as applications would differ from this original depiction.

Clock Services Initialization Use Case

Description

In this use case, clock services initializes and synchronizes the clocks on all systems that will support a test. The TENA application assets introduced here will appear again in the time management use cases.

Actors

- User Application (UA) - this is the appliation that initiates the test
- Distribution Services 1 (DS1) - TENA Distribution Services on machine where UA is loaded
- Clock Services 1 (CS1) - TENA Clock Services on machine where UA is loaded
- Live Attack Aircraft Bridge (AC) - TENA application that receives signals from actual aircraft. It publishes the appropriate TENA classes and subscribes to the appropriate TENA classes, translating these into signals for the aircraft
- Distribution Services 2 (DS2) - TENA Distribution Services on machine where AC is loaded
- Clock Services 2 (CS2) - TENA Clock Services on machine where AC is loaded
- HLA Bridge Federate (BF) - A TENA application that provides an interface between the HLA RTI and the TENA infrastructure. In HLA time management terms, this federate is coordinated and paced.

Run-Time Infrastructure (RTI) - High Level Architecture (HLA) RTI

Enemy Aircraft Simulation (SIM) - A logical time, analytical, air campaign

- where BF, RTI and SIM are loaded
- Clock Services 3 (CS3) - TENA Clock Services on machine where where BF, RTI and SIM are loaded

none

Narrative

UA sets the current time on it's instance of the infrastructure.

UA requests synchronization of all infrastructures involved in the test.

Description

This use case is an example of a viewer and data logger asset replaying data

between a viewer and data logger are shown.

Assumptions

Test plan has been initiated in the usual way .

All appropriate publish, subscribe, and class instantiations have occurred.

represents time stamped data that is replayed) and one event class is added: Replay (a class which transmits control data from a viewer to the

Actors

- User Asset (UA) - this is the asset that initiates the test
- Distribution Services (DS) - TENA Distribution Services
- Data Logger/Delogger (DL) - this asset contains data (most likely filtered) that is time-stamp ordered
- Viewer (V) - this asset provide the user a user interface for viewing the data that will be replayed with the DL, user commands for this asset include those associated with a tape device (rewind, fastforward, play, stop, etc.)
- Clock (C) - this actor is shown to aid in the understanding of the relationship between the wall clock time of the TENA infrastructure and the time-stamps of the replayed data. In the event trace, this actor's line will show the wall clock time of the TENA infrastructure but no interactions with other actors are needed

Narrative

A. Setup for the test plan has occurred through scheduling services, Initialization Manager, and Execution Manager. All completed successfully and the test plan is ready to begin processing.

B. V requests that DL begin playing data at a time-step rate paced with real-time. Note that although the wall clock and time stamp time differs, the two progress forward in time at the same rate.

C. V requests that DL begin playing data at a time-step rate that is four times real-time. In this example, the logger/delogger sends four seconds of prerecorded data for each one second of elapsed wall clock time. Note that the data is sent at a rate that is linear with wall clock time.

D. V requests that DL rewind 5 minutes of data. Note that the time of the time stamps is five minutes earlier and that the rate is the same rate that was set at step 'c' of the narrative (4:1).

E. V requests that DL begin playing data at a time-step rate that is one-quarter real-time. In this example, the logger/delogger sends 1 second of prerecorded data for every four seconds of elapsed wall clock time. As in step 'c', the data is sent at a rate that is linear with wall clock time.

F. The test plan shuts down through the usual procedures.

Event Trace

Infrastructure Support Objects

definition for and instances of classes of basic objects required for system functioning. The set of basic type definitions and associated methods which are

for more complex classes which are needed widely or are fundamental to capabilities supported directly by the range infrastructure. It is also likely that a

the system.

Application Programs

Mandatory Application Programs

Description:

The Network Manager (NM) is a required application asset which is responsible

range operations. It is charged with managing the connection of network elements to form connections, it monitors network and connection performance,

applications. The NM may be a single stand-alone asset or may be integrated into a larger facility management application depending on requirements at each

The NM is intended primarily to deal with the dynamic acquisition and return of communications resources to satisfy individual test requirements. It manages

supported by the operating platforms they are attached to. However, these permanent networks, which support information transfers managed by Message

circuits and networks intended for dedicated public use managed by Connection Services and for private connections directly connecting two specific assets.

communications resources in response to the definition of logical ranges for tests or in response to unplanned user requirements. The NM has information about

facility. In response to requests from AMA to acquire a communications resource with a specified quality of service (QOS) the NM will determine which resource

known to it and available most closely matches the QOS requirements. It then performs the necessary actions to acquire control of the resource. This may involve requesting personnel to connect hardware, configure networks through autonomous network control terminals, requesting and negotiating for common carrier resources, etc. When the NM has determined the resource is available for use by the range facility it reports the status and actual QOS characteristics, along with the input/output channel connection point and machine, to AMA.

During operation the NM monitors all communications resources which it has available to it or knows about and reports this status for use by other facility assets. It may also be augmented with the capability to act to relieve congestion in circuits. AMA may also request that a resource be released when it is no longer required. The NM provides capabilities to accomplish this in a timely manner.

The resources managed by the NM are intended to satisfy all facility requirements and not just computer data transfer. The NM may be required to provide and manage resources for the transport of integrated voice, video, and data communications. It may further be required to provide for the transport of raw, compressed and/or secure voice and video communications. The transport must enable real-time reconstruction of the data by the receiving asset and meet network command/control requirements.

The NM must be extensible and flexible so that it can quickly adapt to new communications technologies as they become available and additional or changes requirements for supporting tests and facility business operations.

Rationale:

It is recognized that processing power is doubling about every two years. LAN capacity development has not kept pace with this increase in processing power resulting in fewer computers being attached to a LAN segment. To meet the increased networking demands new families of LANs / WANs are evolving.

Networking protocols developed ten or more years ago placed greater functionality within network components to compensate for reliability and limitations in data links and to off-load end-point systems. These new emerging transport technologies coupled with more reliable transport media have greatly reduced the need for these embedded network services.

Range communication requirements are viewed as critical design criteria. One of the design goals of the TENA system architecture is to provide the data transport flexibility needed to maximize communication performance and efficiency.

Modern range systems require the integration of voice, video, and data. To meet these demands and the continuing increase in network bandwidth requirements, emerging transport technologies are greatly reducing the services provided by the network elements and moving them to the customer premises equipment. The TENA infrastructure must support both the current generation of transport

services and the emerging networking systems support services. To replace this

infrastructure and logical range support applications provide greater support for network management services.

interconnect range systems. Many point-to-point leased lines have no backup resulting in reliability problems. Additionally, utilization of leased lines may be low

A better networking solution to meeting range communications needs is to reengineer these existing LAN/WAN networks into a new network which provides

expensive lines. This new network architecture allows dynamic capacity allocation to maximize available data bandwidth utilization.

relaying traffic as quickly as possible using fast packet relay or fast packet switching. The networking technology used to transport data is transparent to the

permitting rangers to use the technologies that best meet local requirements.

Capabilities

routing, circuit service, and circuit status. Circuit routing includes those services needed to determine, create, and configure a connection. Circuit service services

switch over operations, and timing control. Circuit status information pertains to the status of a connection/network and status of network elements (assets).

subsets (which may not necessarily coincide with range segments). Each network subset has an associated network manager. Each network manager is

(OAM&P) of its associated network elements.

The OAM&P services provided by TENA network elements allow for traffic

the transport layer in the network elements where a circuit is not able to meet a stated or negotiated level of performance. Traffic control defines a set of actions

measures to adapt to unpredictable fluctuations in traffic flows (information bandwidth) and other problems within the network.

Associated with a connection are certain quality of service attributes. These

connection. The infrastructure and network manager allow for scheduling of network assets with specified QOS parameters and for dynamic redefinition of

QOS during a test. These QOS attributes determine what connection admission control actions and circuit performance parameters will be used in network manager.

It is the policy of the TENA not to manage network QOS but to provide the services that allow user assets the ability to manage. QOS management is shared among the user, the network manager, and the network. The mechanism by which this management is carried out is the service contract. The network user is responsible for agreeing to a service contract with the TENA that stipulates the rules on the use of a network (such as bandwidth); circuit performance requirements (such as circuit reliability); and acceptable alternative actions if degraded conditions exist (such as increased error rates resulting in reduced effective data bandwidth). The network assumes the responsibility of supporting the other QOS requirements. The TENA may support contract re-negotiation to enable dynamic network management.

QOS parameters are defined in the data dictionary. Any particular facility may support all QOS parameters or only a subset of them. They may include the following specifications:

- A QOS parameter which provides for protocol selection. Protocol selection applies to message services as well as to connection services.
- A QOS parameter or parameters which specify the communications latency on a connection.
- A QOS parameter or parameters which specify the required level of reliability of data delivery. The implementation of delivery reliability may be through the protocol selected (TCP/IP or UDP/IP, for example) and/or through the transport technology used.
- A QOS parameter or parameters which specify the number of errors tolerable or error rate limit desired.

Additional QOS parameters may be defined by range system implementers to satisfy local requirements. Each facility will establish a default set of qualities of service which satisfy basic requirements and map to well known, platform supported resources.

The status of network subsets becomes the shared responsibility between caretaker(s) and network manager. Network status is composed of, schedule status and current operating state.

The responsibility for operating, administering, maintaining, and provisioning of range connections lies with the network manager application. The network manager resides outside the infrastructure to allow for tailoring of its services to specific network requirements and special range operations needs. The network manager, in collaboration with network asset caretakers, is required to support

infrastructure scheduling services as well as general network operations support.

that the route that data takes through a network isn't important as long as the connection's QOS is satisfied. The QOS of a connection makes up one side of

The details of the service contract will be determined by the range development group. The procedure used to establish this contract can be described though. In

connections needed to support a test. These requirements are translated to numbers of connections with associated quality of service which drive the NM to

within the test plan and are, in essence, the service contract specification. During test conduct, these connection specifications are extracted from the test plan by

connection requests and passes the connection information along with QOS specifications to the NM.

Services provided:

Reserve Asset	Get Schedule
Clear Schedule	Remove Asset
Process reservation	Query Asset Status
Acquire Asset	Relinquish Token
Get Reservation Queue	

Description

Manages requests for use of assets in the range catalog. An exercise is

catalog are managed here. Range caretakers at each physical range are queried by Asset Manager for the assets which are physically located at that physical

management applications.

Scheduling certain assets may involve human interaction to schedule. This

effects the scheduling and record pertinent information for subsequent use during exercise execution. Asset Manager works in concert with execution services to

happen.

Asset Manager is also responsible for managing access to assets during the execution of a test. Scheduling an asset for use during a test does not guarantee that the asset is available at the scheduled time. The asset may be down for maintenance, another user may still be using it, or any of a number of causes could lead to the unavailability of an asset at the previously scheduled time. During the execution of a test Execution Manager will query Asset Manager when access to an asset is required.Asset Manager will grant access to the asset if the requesting test holds a valid reservation and the asset is available. If access is not granted Execution Manager can query an appropriate range management application for resolution of the conflict.

Additionally, Asset Manager is responsible for the instantiation of instantiable assets which are not currently executing and coordinates with the Network Manager to have dedicated circuits established and allocated.

Services Provided:

Reserve Asset

Description

This service is used to request dedicated service over some period of time from a system asset.Asset Manager will assign a unique request ID to the request and place the request on the asset's request queue.

In Parameters

- Asset ID : Asset_Identifier - the identifier of the asset which the requester wishes to reserve.
- Start Reservation : Date_Time - the date and time at which the reservation is to start.
- Duration : Time - the time of duration of the exercise at which the reservation is planned to expire.
- Request Void: Date_Time - the date and time after which the request is no longer valid.

Out Parameters

- Reservation Request ID : Request_ID - the request identification of the request.

Exceptions

- Asset Does Not Exist -the asset identifier is not a valid identifier.
- Asset Not Available - the asset is not available for scheduling during the entire period requested.

- Caretaker Not Available - the asset's caretaker cannot be reached.
- Invalid Date Time - the start date is invalid or the void time does not follow the start date and time.
- Invalid Duration - the duration is less than or equal to zero or greater than the maximum allowed duration.

Preconditions

- The asset exists.
- The asset's caretaker is up and running.

Postconditions

- The request has been placed on the appropriate caretaker's request queue.

Release Asset

Description

Used by a system component to release an asset which it has previously reserved for use. Returns the reservation token.

In Parameters

- Asset ID : Asset_Identifier - the identifier of the asset which the requester wishes to cancel the reservation for.
- Reservation Token : Asset_Token - the reservation token for the requested asset.

Out Parameters

none

Exceptions

- Asset Does Not Exist -the asset identifier is not a valid identifier.
- Invalid Token - the token provided is not a valid token for the indicated asset or the token is already in possession of the asset's caretaker.
- Caretaker Not Available - the asset's caretaker cannot be reached.

Preconditions

- The asset exists
- The service requester has previously reserved the asset indicated.
- The asset's caretaker is running and reachable.

Postconditions

- The asset reservation is canceled.

Get Schedule

Description

This service is used to obtain the current schedule for an asset or set of assets managed by a caretaker.

In Parameters

- Asset ID : Asset_Identifier - the identifier of the asset.
- Mode : Asset_Query_Mode - this parameter is null if the asset of interest is not a caretaker. If it is a caretaker the mode indicates if requester desires the asset's own schedule or the schedule for all assets managed by it.

Out Parameters

- Schedule : Asset_Schedule_Set - a reference to a set (1 or more) of asset schedule aggregations.
- Number Records : Integer - the number of asset aggregations returned. Set to zero if request fails.

Exceptions

- Asset Does Not Exist -the asset identifier is not a valid identifier.
- Caretaker not Available - the asset's caretaker cannot be reached.
- Asset Not Caretaker - requested a set of asset schedule records from an asset which is not a caretaker.

Preconditions

- The asset exists.
- The asset's caretaker is running and reachable.

Postconditions

Clear Schedule

A service called to mark an asset's schedule as completely open. Only the asset's caretaker can clear its schedule.

-

Out Parameters

- Schedule Cleared : Boolean - indicates whether schedule was successfully cleared.

-
-
-

Preconditions

- The asset exists.
- The asset's caretaker is running and reachable.

Postconditions

The asset's schedule is marked completely open if the asset of interest is

of interest is marked completely open if the asset of interest is a caretaker.

Add Asset

Description

This service is used to add a new asset to the system. This includes adding the asset to a caretaker's set of managed assets, modifying the local asset catalog and schedule to reflect the new asset, and adding the new asset to the master range asset catalog.

In Parameters

- Asset Name : String - the name of the asset to be added.
- Asset ID : Asset_Identifier - the identifier of the asset.
- Caretaker ID : Asset_Identifier - the asset identifier of the caretaker which is to manage the new asset.
- Schedule Mode : Asset_Schedule_Mode - the scheduling options which are allowed for the asset (exclusive, shared/unlimited, shared/limited).
- Number Tokens : Integer - if the schedule mode is shared/limited then the number of tokens to generate for the asset is indicated by this parameter. Must be => 1.
- Asset Descriptors : Asset_Descriptor_List - an aggregation of asset descriptor pairs consisting of asset descriptors and their values.

Out Parameters

None

Exceptions

- Duplicate Asset ID - the asset identifier supplied for the asset is not unique
- Asset Does Not Exist -the asset identifier of the caretaker asset is not a valid identifier.
- Caretaker Not Available - the caretaker cannot be reached.
- Could Not Add - the service could not be performed because the local

asset catalog could not be modified, the master range asset catalog could not be updated, or the local schedule could not be updated.

Preconditions

- The caretaker is running and reachable.

Postconditions

- The master range asset catalog is updated with the new asset.
- The local asset catalog at the designated caretaker is updated with the new asset.
- The local schedule at the designated caretaker is updated with the new asset.

Remove Asset

Description

This service is used to remove an asset from the system. This includes removing the asset from its caretaker's set of managed assets, modifying the local asset catalog and schedule to reflect the asset deletion, and deleting the asset from the master range asset catalog. Only an asset's caretaker can remove it.

In Parameters

- Asset ID : Asset_Identifier - the identifier of the asset.

Out Parameters

none

Exceptions

- Asset Does Not Exist -the asset identifier of the asset is not a valid identifier.
- Caretaker Not Available - the caretaker cannot be reached.
- Unable To Remove - the asset removal could not be completed.
- Not Caretaker - the requester is not the assets caretaker.

Preconditions

- The asset exists.
- The asset's caretaker is running and reachable.

Postconditions

- The asset is removed from the master range asset catalog, the local asset catalog, and the local asset schedule.

Modify Asset Property

Description

This service is used to modify the value of an asset's property. These properties are listed in the asset catalogs and used to control use of the asset. Only an assets caretaker can modify it's properties.

In Parameters

- Asset ID : Asset_Identifier - the identifier of the asset.
- Property : Asset_Property - the name of the property which is to be changed.
- Value : Asset_Property_Value - the new value of the property.

Out Parameters

None.

Exceptions

- Asset Does Not Exist -the asset identifier of the asset is not a valid identifier.
- Caretaker Not Available - the asset's caretaker cannot be reached.
- Unable To Change - the asset's property value could not be changed.
- Invalid Property - the property named is not an asset property.
- Not Caretaker - the requester is not the assets caretaker.

Preconditions

The asset exists.

The asset's caretaker is running and reachable.

- catalog and in the local asset catalog.

Query Asset Status

Description

asset's properties.

In Parameters

Asset ID : Asset_Identifier - the identifier of the asset.

- property values.

Exceptions

Asset Does Not Exist -the asset identifier of the asset is not a valid

-

Preconditions

- The asset exists.

- The asset's caretaker is running and reachable.

Postconditions

None.

Acquire Asset

Description

Users invoke this service to obtain use of an asset. That use may be exclusive or shared depending on the asset. Once acquired an asset is available for use by the requester until the asset is relinquished. If the asset is an instantiable asset, Asset Manager is responsible for instantiating it and generating an instance ID. If the asset is a dedicated communications channel, Asset Manager is responsible for coordinating with the Network Manager to have the dedicated circuit established and allocated.

In Parameters

- Asset ID : Asset_Identifier - the identifier of the asset.
- Reservation Token : Asset_Token - the reservation token for the requested asset

Out Parameters

- Access Token : Asset_Token - the access token conferring the right to access and use a particular asset. The null token is returned if an access token is not available. This can occur if a component has acquired the asset but not yet relinquished it or if the reservation token does not cover the time the request is made.

Exceptions

- Asset Does Not Exist -the asset identifier of the asset is not a valid identifier.
- Caretaker Not Available - the asset's caretaker cannot be reached.

Preconditions

- The asset exists.

- The caretaker is running and reachable.
- The requester holds a reservation token for the asset.

Postconditions

- The requester has guaranteed access within the limitations for the asset to use of the asset until relinquished.
- If the asset is an instantiable asset an instance of the asset is instantiated.
- If the asset is a dedicated communications channel the channel is connected and ready for use.

Relinquish Asset

Description

This service is used to relinquish an asset and make it available for use by other applications. If the asset is an instantiable asset, Asset Manager will terminate the instance of the asset. If the asset is a dedicated communications channel, Asset Manager will coordinate with the Network Manager to disconnect the dedicated channel.

In Parameters

- Asset ID : Asset_Identifier - the identifier of the asset.
- Access Token : Asset_Token - the access token conferring the right to access and use a particular asset.

Out Parameters

None.

Exceptions

- Asset Does Not Exist -the asset identifier of the asset is not a valid

identifier.

- Caretaker Not Available - the asset's caretaker cannot be reached.

Preconditions

- The requester holds an access token for the asset.

Postconditions

- The access token is available for acquisition by a component.
- If the asset is an instantiable asset this instance of the asset is terminated.
- If the asset is a dedicated communications channel the channel is disconnected.

Relinquish Token

Description

This service is a callback service invoked by Asset Manager on a holder of a reservation token. It is exercised when a scheduling conflict arises from a high priority user requesting to override an existing reservation or allocation for an asset.

In Parameters

- Asset ID : Asset_Identifier - the identifier of the asset which holds the token.

Out Parameters

None.

Exceptions

- Asset Does Not Exist -the asset identifier of the asset is not a valid

identifier.

Caretaker Not Available - the asset's caretaker cannot be reached.

Relinquish Failure - the asset failed to return the reservation token.

-

Postconditions

The token is returned to the asset's caretaker.

Description

This service is used to extract the current set of pending reservation requests for

In Parameters

- Asset ID : Asset_Identifier - the identifier of the asset whose reservation request queue is desired.

  Mode : Asset_Query_Mode - this parameter is null if the asset of interest

  desires the asset's own request queue or the request queue for all assets managed by it.

-

  asset scheduling request aggregations.

- NumberRecords : Integer - the number of asset scheduling request aggreations returned.

-
-
-

for an asset which is not a caretaker

Preconditions

- The asset exists
- the asset's caretaker is running and reachable.

Postconditions

None

Process Reservation

Description

This service is used by a caretaker to command that a pending asset reservation be processed. Processing a request implies that it is either being satisfied or denied. Satisfying a request involves the recording of the requesters reservation and dispatch of the reservation token to the requester. Denying a request implies the dispatch of a null reservation token. In both cases the request is deleted from the request queue and the Reservation Token is delivered to the reservation requester.

In Parameters

- Reservation Request ID : Request_ID - the request identification of the reservation request to be processed
- Satisfy : Boolean - set to true if the request is to be satisfied.

Out Parameters

None

Exceptions

- Reservation Request Does Not Exist - the reservation request identifier is not valid
- Not Caretaker - the requester is not the caretaker of the asset requested in the reservation request

- the requester.

Preconditions

The reservation request exists.

-
-
- requester if the reservation was satisfied. If the reservation was not satisfied a null token was sent.

Rescind Reservation

This service is used to rescind a pending reservation request. It must be called by the application which originally issued the request. In response to this service

reservation request queue.

In Parameters

Reservation Request ID : Request_ID - the request identification of the

Out Parameters

- Request Rescinded : Boolean - indicates whether request was successfully removed from the request queue.

- not valid. Either the request was never made, has expired, or has already been serviced.

- Not Requester - The asset requesting that the reservation request be rescinded is not the original issuer of the request

Preconditions

- The request has been issued.
- The request has not yet been serviced
- The request has not expired.

Postconditions

- The request is removed from the appropriate reservation request queue.

Execution Manager

Services Provided:

Run Plan	Terminate Plan	Pause Plan
Resume Plan	Replace Asset	

Description

Execution Manager (EM) manages the execution of a test or exercise plan on a logical range. Based on the set of assets and events defined in the plan, EA acquires the assets, connects them up as planned, brings them to an initialized state, ensures they are ready to begin execution of a plan, and shuts them down in an orderly fashion upon completion or termination of the plan. In the event that the plan must be interrupted, this service manages the pause and resume of a plan. Execution Manager supports the dynamic replacement of assets in an executing plan. A plan can be validated prior to its actual execution by running the plan in validate mode.

Assumptions

A Plan Repository(PR) exists and is accessible by Execution Manager. The PR

The management of plans executed by this service is outside the scope of Execution Manager. It is assumed that a mandatory application exists, Planner

modifying and deleting of plans from the Plan Repository.

Execution Manager can execute multiple plans simultaneously as long as the

- Execution Manager
  this sort of interaction or human assistance and provides mechanisms for the assets to be secured for use. The status of these assets must be

  Manager
  it that it needs.

Services Provided

Description

Range management applications make use of this service to request that a

is defined in the test or exercise plan to be instantiated. This act involves the acquiring of all the assets which make up the logical range, the instantiation of

directly involved in the actual conduct of a test or exercise.

Rationale

- be executed.
- Mode: PlanExecutionMode - this parameter indicates the mode of execution for the identified plan. If it is null, then normal plan execution will
  will parse the

  instantiate, or initialize assets.

Out Parameters

None.

- Plan Not Found - the indicated plan does not exist.
- Invalid Plan ID - the indicated asset is not a plan.
- Invalid Plan:Reason - the indicated plan is not valid; the associated reason parameter supplies details on the invalid portion of the plan.

Preconditions

- The plan has been added to the Master Asset Catalog and assigned a unique Asset ID.
- Assets listed in the plan are available.

Postconditions

- The logical range is configured, instantiated, initialized, and ready to begin the test.

PausePlan

Description

This service is used to cause a pause to be sent to the range assets which are constituted for a plan. The semantics of a pause to assets are that the asset is to cease producing changes to its internal state when those changes would result from plan play. Sensors would continue to operate and data display assets would continue to display the information they currently possess.

In Parameters

- Plan ID : Asset_ID - the Asset ID of the asset which contains the plan to be paused.

Out Parameters

- None.

Exceptions

- Plan Not Active - the indicated plan is not currently executing.

Preconditions

- The plan has been successfully instantiated through the RunPlan service.

Postconditions

The plan has been paused.

Description

This service is used to cause a resume to be sent to all range assets which have

and respond to events and data as they are defined to.

In Parameters

Plan ID : Asset_ID - the Asset ID of the asset which contains the plan to

Out Parameters

- none.

Exceptions

Plan Not Paused - the indicated plan is not currently paused.

Plan Not Active - the indicated plan is not currently being executed.

- currently paused.

Postconditions

The indicated plan has resumed execution.

TerminatePlan

Description

This service is used to cause the orderly termination of a plan. Assets which must engage in special processing to effect an orderly and error free shutdown are directed to do so. Input/output devices and channels are flushed and shut down, files are closed, and access tokens for assets are returned to their caretakers. Instantiable assets are terminated.

Rationale

In Parameters

- Plan ID : Asset_ID - the Asset ID of the asset which contains the plan to be resumed.

Out Parameters

- none.

Exceptions

- Plan Not Active - the indicated plan is not currently being executed.

Preconditions

- The indicated plan has been successfully placed in execution.

Postconditions

- The plan is no longer executing and the logical range has been dissolved.
- Asset access tokens have been returned to the asset caretakers.

ReplaceAsset

Description

This service is used to substitute one asset for another in an executing plan.

Run-time modification of a plan is needed to support events such as asset failure. In such an instance, it may be possible to remove the failed asset and

offending asset for the remainder of the plan's execution or until which time the Test Conductor chooses to remove it from the plan execution.

-
-
-

  be resumed.

Out Parameters

none.

Exceptions

- Plan Not Active - the indicated plan is not currently being executed.
- Plan Not Paused - the indicated plan is not currently paused.
- InvalidAsset:AssetID - either the existing asset ID is not valid for the identified plan, or the replacement asset is not available or a legitimate replacement

Preconditions

- The indicated plan has been successfully placed in pause.

Postconditions

- The plan has been modified and the existing asset has been replaced in the associated plan.
- The plan is still in pause mode.
- The permanent copy of the plan is NOT modified.

Initialization Manager

Description:

Initialization Manager manages assets used to initialize other assets associated with a Test Range. These assets are typically data sets used by applications to establish an exercise's initial conditions, e.g., terrain databases, environment settings, and start conditions. Initialization assets can also include bootstrap procedures for devices, interactive scripts that require a human-in-the-loop, or any other entity that supports initialization. These assets may or may not be associated with a test plan.

Rationale

The variety of initialization assets and their distributed location suggests the need for a unified concept and generalized service to access initialization assets. Without this service the applications will be forced to implement access mechanisms for each type. The existence of such a service also maximizes reuse potential since available initialization assets will be made visible to the community at large for use in a particular exercise or as examples for use in a similar context.

Assumptions

The following assumptions are made regarding Initialization Manager:

- Life cycle management (e.g., create, destroy) of initialization assets is outside the scope of the service. This service is focused on managing access to these assets.
- Any initialization asset can be included in the range asset catalog. Whether it is actually catalogued is dependent on the creator of the asset. However, any initialization asset referenced in a test plan must be included in the range asset catalog.

- Initialization assets have important relationships with other assets that are retained as part of the asset catalog. For instance, an initialization data set

  value to another type of asset. These relationships can be defined at the asset level (e.g., test plan T uses initialization data set D) or at the

- being initialized. For instance, if three assets are required for a test, and each requires initialization data, then the data is partitioned into three

  assets that provide broader initialization than what is required for the asset thereby promoting reuse.

  There is a many-to-one relationship between an asset type and

- and used in multiple instances of assets that require that type of initialization.

  An initialization asset exists at various states of readiness. These states

  explanation).

Service Context

processing is depicted in the following diagram and each entity is described here:

- Distribution service (DS) - the infrastructure service that manages the distribution of data among range assets

  Asset Manager (AM)- the infrastructure service that manages the

- Manager (IM) will maintain a persistent store of initialization asset information. This persistent collection is given the name of Initialization

- that requests an initialization service.

Services Provided:

The following services are specified for Initialization Manager:

- AddInitAsset - introduces an initialization asset entry into the IAC
- RemoveInitAsset - deletes an initialization asset entry from the IAC
- GetIAContent - retrieve the content of a specified initialization asset
- CopyIAContent - place a copy of the initialization asset contents into another asset specified by the requesting asset
- ModifyIAAccessProfile - change the value of one or more access properties associated with an IA entry in the IAC
- GetIAProfile - get a copy of a selected IA profile for an IA entry
- ReleaseIAContent - destroy instances of IA content exchange classes

AddInitAsset

Description:

This service provides for adding an initialization asset entry into the IAC. An initialization asset is added into the IAC after it has been assigned an asset ID and added into the Master Asset Catalog(MAC) through Asset Manager. One of two IA entry types can be added to the IAC. An entry can include both its metadata and content. In this case, a content parameter is supplied that contains a list of ICC static class and their attribute values. An entry can also be a pointer to the actual content. In this case, an access profile is provided.

Rationale:

While Asset Manager provides the capability to add assets to the Master Asset Catalog, there is a need to retain additional information about an initialization asset. For example, the access properties will provide details of where the asset is located and the method of access for subsequent retrieval and copy.

In Parameters:

IA_ID: AssetID - unique identifier for IA as generated by Distribution services

IAContentExchangeProfile: <sequence>ICCClass - Contains a list of ICC static classes used for exchanging IA content through distribution services. Static classes are used as opposed to event classes since the class instances must persist beyond a service call (see the retrieve and copy use cases).

IAContent: <sequence>IAContentDescriptor - Contains a list of ICC class IDs and a corresponding set of attribute values. All records collectively constitute the content of IA. Will be null if the IA content is not included.

IAAccessProfile: <sequence>IAAccessDescriptor - Contains a list of properties and their values used for accessing an initialization asset (e.g., location, format, protocol, access product description, query parameters). Will be null if the IA content is included.

Out Parameters

None

Exceptions:

InvalidParameter: ParameterType - invalid value supplied for an input parameter; identifies parameter at fault through ParameterType argument

DuplicateDescriptor - the supplied combination of name and type is not unique

IncompleteMandatoryData(missing data) - all required data not provided in service call; the attribute identifies the type of data missing.

NotInMAC - identified asset is not known to Asset Manager

Preconditions

Asset exists

Asset has been successfully added to the MAC

Post Conditions

- Asset is known to the infrastructure and can be used in carrying out test range activities via the infrastructure (e.g., can be assigned as an initialization asset to another asset in a test plan).

Interface:

RemoveInitAsset

Description:

Remove an initialization asset entry from the IAC. Note that this service does not delete the initialization asset; it merely makes it unavailable through the infrastructure.

Rationale:

An initialization asset may need to be removed from the IAC for a number of reasons: using assets have been retired from the range, it is no longer a correct initialization data set or procedure,…

In Parameters

IA_ID: AssetID - unique identifier for IA

Out Parameters

None

Exceptions

InvalidParameter: ParameterType - invalid value supplied for an input parameter; identifies parameter at fault through ParameterType argument

NonExistent - no such asset is known to the infrastructure

Preconditions

Asset has been previously added to the IAC

Post Conditions

Asset is no longer available through Initialization Manager.

Interface

GetIAProfile

Description:

Retrieve a copy of a selected profile type from an IA entry

Rationale:

In order to provide tools like catalog browsers, it is necessary to provide a service that can retrieve the profiles contained in an IA entry.

In parameters:

profiles for All IA entries are requested

IAProfileType:ProfileType - describes the type of profile information to be

will be returned.

Out Parameters

properties and their values for the referenced IA (May be null).

IAContentExchangeItems:<sequence>ICCClass - provides a list of classes used

Exceptions

InvalidParameter: ParameterType - invalid value supplied for an input parameter;

NonExistent - initialization asset is not known to the infrastructure

InvalidePropertyType - invalid value for property type

IA has been successfully added to the IAC

Post Conditions

Interface

ModifyIAAccessProfile

Assign values to the access properties of an initialization asset

Rationale:

there is likely to be a need to change or extend values required for access to the asset. For instance, an initialization asset could be moved to a different device or

location or its visibility to infrastructure users may be temporarily restricted. Note that only access profile information can be modified with this service; modification of content must be performed outside the infrastructure or through a remove/add combination of services.

In Parameters

IA_ID: AssetID - unique identifier for IA

IAAccessProfileItems: <sequence>IAAccessDescriptor - a list of one or more access properties and their corresponding values that need to be changed.

Out Parameters

none

Exceptions

InvalidParameter: ParameterType - invalid value supplied for an input parameter; identifies parameter at fault through ParameterType argument

NonExistent - initialization asset is not known to the infrastructure

InvalidProperties - the IA referenced does not have the properties provided

Preconditions

IA has been successfully added to the IAC

Post Conditions

IA access profile information has been successfully updated in the IAC

Interface

GetIAContent

Description:

Provide the content of an initialization asset to a requester.

Rationale:

This service provides a common mechanism for retrieving initialization data regardless of the location or underlying storage mechanism of the asset. This

In Parameters

IA_ID: AssetID - unique identifier for IA

IAContent - actual value of IA is supplied to subscriber of the IAContent Exchange classes

InvalidParameter: ParameterType - invalid value supplied for an input parameter; identifies parameter at fault through ParameterType argument

QoSUnavailable - QoS required is not currently available

Preconditions

Distribution services is executing; IA depends on the publish/subscribe services within DS to perform its copy and retrieve services

Requesting asset has acquired the content of the initialization asset

Interface

CopyIAContent

Description:

Rationale:

There are several reasons for applications to acquire a copy of an initialization

request a preload of the initialization data, e.g., a terrain database; test execution performance may dictate the co-location of an initialization asset with the asset

itself; a new asset's initialization asset can be derived from an existing initialization asset; to debug a test plan it may be desirable to load a copy of the initialization data to the debug environment, …

In Parameters

IA_ID: AssetID - unique identifier for IA to be copied

CA_ID: AssetID - a unique identifier for the asset that will receive a copy of the contents of the requested IA

Out Parameters

IAContent - actual value of IA is supplied to subscriber (which in this case is an instance of this service, IA) of the IAContent Exchange classes

Exceptions

InvalidParameter: ParameterType - invalid value supplied for an input parameter; identifies parameter at fault through ParameterType argument

NonExistent - initialization asset or copy asset is not known to the infrastructure

QoSUnavailable - QoS required is not currently available

Preconditions

IA has been previously added to the IAC

CopyAsset has been acquired from AMA by the requesting asset

Post Conditions

Contents of IA are contained in CopyAsset

Interface

ReleaseIAContent

Description:

Destroy all instances of IA content exchange classes for the identified initialization asset

Rationale:

instantiated for purposes of retrieving the content of an IA. Typically used by AMA when Execution Manager relinquishes an initialization asset.

IA_ID: AssetID - unique identifier for IA as generated by Asset Manager

Out Parameters

Exceptions

InvalidParameter: ParameterType - invalid value supplied for an input parameter;

NonExistent - initialization asset is not known to the infrastructure

NotReleasable - no instances exist for release

A GetIAContent service has been previously requested for this IA

Post Conditions

Interface

Use Cases

It should be noted that the following use cases and event traces were developed

considered Service Groups within the Infrastructure and not as required applications supporting the infrastructure. Although these use cases are presented in their original form, the reader should be aware that updated versions depicting these three capabilities as applications would differ from this

To illustrate and validate the execution of Initialization Manager, the following set of use cases is provided.

Add an initialization asset

Remove an initialization asset

- Retrieve an initialization asset
- Copy an initialization asset
- Get an IA entry from IAC
- Modify IA access profile

Add an Initialization Asset

Description:

This use case shows how an initialization asset is added into the infrastructure. A transaction is defined to ensure that the asset is added to the MAC before its inclusion in the IAC.

Actors:

- Initialization Asset Collection(IAC) - the logical collection of initialization asset information needed for retrieval and updates by the infrastructure.
- Requesting asset (RA) - the asset that is requesting that IA be added to the IAC.
- Initialization Manager (IM) - infrastructure service that manages initialization assets and makes them available to system components
- Distribution Services (DS)- infrastructure service used to publish and subscribe to assets.
- Asset Manager (AM)- infrastructure service used to schedule assets.

Scenario :

Only one scenario is illustrated even though there are two ways to store an initialization asset entry into the IAC. The only difference in the two event traces is a slightly different calling sequence on AddInitAsset (see service description).

Assumptions

- Requesting asset (RA) is executing.
- The initialization asset type has been defined

Scenario Steps

a) RA sends an event, GenerateAssetIDRequest, to DS.

b) DS sends an event, GenerateAssetIDResponse, to RA supplying a unique asset ID, IA_ID.

c) RA sends an event, AddAssetRequest, to AMS via DS

d) DS propagates the AddAssetRequest to AMS where it is added to the master asset catalog.

e) RA sends an event, AddInitAssetRequest, to IS via DS.

f) DS propagates the AddInitAssetRequest to IS.

g) IS updates the IAC entry for IA

Event Trace:

Remove an Initialization Asset

Description:

This use case shows how an initialization asset is removed from the infrastructure. A transaction is defined to ensure that the asset entry is removed from the Master Asset Catalog prior to removal from the IAC.

Actors:

- Initialization Asset Collection(IAC) - the logical collection of initialization asset information needed for retrieval and updates by the infrastructure.
- Requesting asset (RA) - the asset that is requesting that an initialization asset entry be removed from the IAC.
- Initialization Manager (IM) - infrastructure service that manages initialization assets and makes them available to system components

- Distribution Services (DS)- infrastructure service used to publish and subscribe to assets.
- Asset Manager (AM)- infrastructure service used to schedule assets.
- Master Asset Catalog (MAC) - the logical collection for holding information about all assets known to the TENA infrastructure.

Scenario:

Assumptions

- IA has been previously added to the IAC and the MAC
- Requesting asset (RA) is executing.

Scenario Steps

a) RA sends an event, RemoveAssetRequest, to AMS via DS
b) RA sends an event, RemoveInitAssetRequest, to IS via DS
c) DS propagates the RemoveInitAssetRequest to IS.
d) IS removes IA entry from the IAC

Event Trace:

<u>Retrieve an Initialization Asset</u>

Description:

This use case shows how the content of an initialization asset is retrieved through the infrastructure.

Actors:

Initialization Asset Collection(IAC) - the logical collection of initialization asset information needed for retrieval and updates by the infrastructure.

Subscribing asset (SA) - an asset that subscribes to the initialization content to be published by IS

Initialization Manager (IM) - infrastructure service that manages initialization

Distribution Services (DS)- infrastructure service used to publish and subscribe to assets.

Scenario 1: Retrieve from IAC - in this scenario, the content of the asset was included in the entry to the IAC

Assumptions

IA entry has been added to IAC and the content was supplied during the add

IA entry has only one type of content described by a single ICC class and only one instance of that class as its content

SA is executing and has subscribed to the single ICC IA content exchange class using an agreed upon QoS

SA subscribes to the IA content exchange classes

AMS sends an event, GetIAContentRequest, to IS via DS

contained within the IAC

IS publishes the ICC class contained in the IA entry

entry

DS propagates a DiscoverInstance to SA for the registered instance

IS sends an Update to SA for the instance

DS propagates the update to SA

Scenario 2: Retrieve via IAC - in this scenario, the content of the asset was not

supplied, only an access profile

In order to illustrate the retrieval of an initialization asset not stored in the IAC, it is assumed that IS has the capability to read the IA in its native format and publish its contents according to the IAContent Exchange Profile. In turn, the SA subscribes to these classes to receive the content through distribution services. The basic flow of information is depicted in the following diagram.

Assumptions

IA has been previously added to IAC and the access profile was supplied during the add request

SA is executing and has subscribed to the two IA classes

Scenario Steps:

AMS sends an event, GetIAContentRequest, to IS via DS

IS retrieves the IA entry from the IAC

IS publishes IAClass1 and IAClass2, the two classes defined in the IA Content Exchange Profile

IS uses the access profile information to access IA and perform a native read

IS sends a RegisterInstance to DS for IAClass1 and receives an instance ID

DS sends a DiscoverInstance to SA for the IAClass1 instance

SA sends a RequestUpdate to DS for the class instance which is propagated to IS

IS sends an Update to DS for the class instance which is propagated to SA

IS sends a RegisterInstance to DS for the IAClass2 and receives an instance ID

DS sends a DiscoverInstance to SA for the IAClass2 instance

SA sends a RequestUpdate to DS for the class instance which is propagated to IS

IS sends an Update to DS for the IAClass2 instance which is propagated to SA

Event Trace:

Copy an Initialization Asset

Scenario1: Copy via IAC - IA is located on a different instance of infrastructure

In order to illustrate the copy of an initialization asset not stored in the IAC, the implementation model assumed in the Retrieve use case is extended to include

Furthermore, it is assumed that the copy is to be controlled through a different instance of the infrastructure. In this scenario, IS is both the publisher and

implementation model is illustrated in the diagram below.

Actors:

Initialization Asset Collection(IAC) - the logical collection of initialization

-
- initialization data
- Initialization Manager (IM) - infrastructure service that manages initialization assets and makes them available to system components

  Distribution Services (DS)- infrastructure service used to publish and

-

Assumptions

- IA has been previously added to IAC and the access profile was supplied during the add request

  IA content exchange profile has two classes defined

  IA content contains only one instance of each profile class

  RA is executing on the same infrastructure as the IA controlling instance,

- the ICC)
- DS is executing on both infrastructure instances

- IS implementation has component(s) that can
- read IA in its native format and publish each read as two IA content exchange classes
- subscribe to published IA content exchange classes and write IA in its native format
- Completion status is contained within an IA content exchange class as an attribute
- A copy asset has been acquired and has a unique id, CA_ID
- AMS can supply IS with the infrastructure instance to use to access CA_ID through the asset properties supplied during an AddAsset request.

Scenario Steps:

a) RA sends an event, CopyIAContentRequest, to IS-1 via DS-1

b) IS-1 retrieves the IA entry from IAC

c) IS-1 sends an event, QueryAssetStatusRequest, to AMS-1 via DS-1 to get the correct infrastructure instance to coordinate with for the copy service

d) AMS-1 sends the CA_ID Properties in a QueryAssetStatusResponse which is propagated to IS-1

e) IS-1 sends an event, IACopy to IS-2, supplying it with the CA_ID and the IAEntry

f) IS-2 subscribes to the IAClass1 and IAClass2

g) IS-1 performs a native read on IA

h) IS-1 registers and instance of IAClass1 and IAClass2 which are propagated to IS-2 through DS-1 to DS-2

i) IS-2 sends a RequestUpdate to IS-1 for each class instance

j) IS-1 sends an Update to IS-2 for each class instance

k) Upon receipt of both updates, IS-2 performs a native write to CA

Event Trace:

Notes

Asset states vary along a two-dimensional state space. One dimension represents the maturity of the asset as it is developed, refined and tested. The second dimension represents its visiblity to the TENA infrastructure through its presence in the Range Asset Catalog. The valid states of asset maturity are:

1. NonExistent: This is the starting point, when a need for a new asset has been identified but no steps have yet been taken to develop or purchase the asset.

2. InProcurement: In this state, an asset is being purchased, discovered or developed, but is not yet available to the ranges.

3. Exists: In this state, the asset has been delivered to someone within the range system, but is not necessarily ready for the intended application. For example, the asset may be a commercial product that needs to be tailored to the needs of the ranges.

4. Customized: In this state, any tailoring that is required to support the intended application has been completed.

5. FunctionallyValidated: In this state, the asset has successfully undergone functional tests and is known to operate correctly and to provide the desired capability.

6. TENAComplianceValidated: In this state, the asset has successfully passed all TENA compliance tests. Note that this state may not apply to all assets.

7. Functionally&TENACompliant: In this state, the asset has passed both functional and TENA compliance tests.

In addition to the maturity dimension of an asset, there exists a visibility dimension. The valid states of asset visibility are:

1. Uncataloged: In this state, the asset does not appear in the Range asset catalog. The TENA infrastructure has no visibility into this asset. The asset can not be connected into the logical range.

2. Cataloged: In this state, the asset is logged in the asset catalog. The TENA infrastructure has visibility into the asset. Other users may be able to examine the properties of the asset or schedule use of the asset, depending on the access restrictions.

All assets in the NonExistent state are Uncataloged. Assets may be cataloged in any of the other states. The following graph shows the possible states.

Events transform an asset from one state to another, as is indicated in the following state transition diagrams. Each event is capable of changing only one dimension of the asset's state. So, for example, events that change the visibility of the asset (from uncataloged to cataloged or back) leave the maturity of the

asset unchanged.

The figure below illustrates the visibility dimension of the asset life cycle combined with its maturity and catalog state.

Notes:

- An advantage of early cataloging is to share with other ranges the status of an asset that is under development, even though it may not yet be ready for use.
- Other range assets will likely be needed in order to undergo functional and compliance testing. These assets will be scheduled through the TENA infrastructure in the same way that they would be scheduled and reserved for a customer test.
- An asset may be retired from any state if it is Uncataloged.
- In order to extend a validated asset (i.e. create a new version) a new asset is created in the NonExistent state. The procurement process may involve capturing a copy of the initial asset and modifying it. Once the new version is validated and cataloged, the old version may be retired or may be retained for legacy users.